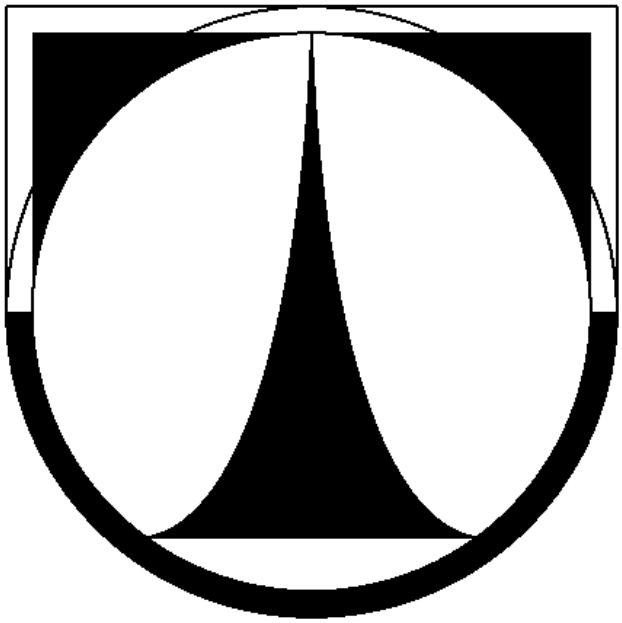
TECHNICKÁ UNIVERZITA V LIBERCI

Ekonomická fakulta



BAKALÁŘSKÁ PRÁCE

2012 **Andrea Mayová**

TECHNICKÁ UNIVERZITA V LIBERCI

Ekonomická fakulta

Studijní program: B 6209 Systémové inženýrství a informatika
Studijní obor: Manažerská informatika

Využití reportovacího nástroje JasperReports v celopodnikových informačních systémech

Usage of the reporting tool JasperReports for the enterprise-wide information systems

BP-EF-KIN-2012-11

Andrea Mayová

Vedoucí práce: Ing. Dana Nejedlová, Ph.D., katedra informatiky
Konzultant: Ing. Petr Motl, OR-CZ spol. s r.o.

Počet stran: 91 Počet příloh: 5

Datum odevzdání: 4. 5. 2012

Zadání ZP

Vložit originál

Zadání ZP

Vložit originál

Prohlášení

Byla jsem seznámena s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. O právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědoma povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracovala samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci, 4. 5. 2012

Poděkování

Tímto bych velice ráda poděkovala vedoucí mé bakalářské práce Ing. Daně Nejedlové, Ph.D. za odbornou spolupráci a velmi cenné připomínky a rady. Děkuji.

Mé díky patří také firmě OR-CZ, spol. s r.o., především mému garantovi praxe a konzultantovi bakalářské práce Ing. Petru Motlovi za trpělivost, odborné připomínky a poskytnutí podkladů pro zpracování bakalářské práce. Děkuji.

Anotace

Tato bakalářská práce se zabývá využitím reportovacího nástroje JasperReports v celopodnikových informačních systémech. Hlavním cílem této práce je vytvoření metodiky tvorby reportů s využitím nástroje Hibernate, s čímž souvisí tematika objektově relačního mapování za pomoci tohoto nástroje, která je obsahem první části této práce. Samotný reportovací nástroj a jeho možnosti při tvorbě reportů je představen v části druhé. Metodika tvorby tiskových sestav byla uplatněna na konkrétních datech uživatelů podnikového informačního systému OR-SYSTEM, což je obsaženo ve třetím bodě této práce. Na základě požadavků na tvorbu sestav od samotných uživatelů OR-SYSTEMu je metodika uzpůsobena tak, aby byla co nejjednodušší a blížila se co nejvíce jejich představám. Poslední část práce popisuje zhodnocení vytvořené metodiky za pomocí nástroje JasperReports a pro srovnání jsou zde popsána i dosavadní řešení tvorby reportů.

Klíčová slova

Objektově relační mapování, informační systém, OR-SYSTEM, Hibernate, report, JasperReports, iReport, Java

Annotation

This bachelor thesis deals with the use of the reporting tool JasperReports in enterprise-wide information systems. The main objective of this bachelor thesis is to create a methodology of creating reports by using Hibernate tool, what is related to the theme of object-relational mapping by usage of this tool, this is contain of the first part of this bachelor thesis. The reporting tool itself and its ability to create reports are presented in the second part. The methodology of creating reports has been applied to specific data of users of enterprise information system OR-SYSTEM, which is described in the third article of the thesis. The methodology is adapted to the requirements for creating reports by the users of OR-SYSTEM, so that it is as simple as possible and most respecting users' expectations. The last part describes evaluation of the methodology developed with using the tool JasperReports and for comparison there are described also current solutions of creating reports.

Keywords

Object-relational mapping, information system, OR-SYSTEM, Hibernate, report, JasperReports, iReport, Java

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódů

Seznam zkratek

API	Application Programming Interface
CSV	Comma-separated values
ERP	Enterprise Resource Planning
HQL	Hibernate Query Language
HTML	HyperText Markup Language
IP	Internet Protocol
JDBC	Java Database Connectivity
JPA	Java Persistence API
JRXML	JasperReports Extensible Markup Language
LGPL	Lesser General Public License
ODT	Open Document Text
ORM	Objektově relační mapování
PDF	Portable Document Format
POJO	Plain Old Java Object
RTF	Rich Text Format
SQL	Structured Query Language
WYSIWYG	What you see is what you get
XLS	Microsoft Excel
XML	Extensible Markup Language

Úvod

V rámci třetího ročníku oboru Manažerská informatika na Technické univerzitě v Liberci jsem měla možnost absolvovat roční řízenou praxi ve firmě OR-CZ, spol. s r.o. v Moravské Třebové. Ve společnosti jsem byla zařazena do divize ERP, konkrétně do oddělení Vývoj.

Po nástupu na praxi jsem byla obeznámena s náplní své činnosti ve firmě, ze které také vychází tato bakalářská práce. Touto náplní bylo především vytvoření metodiky pro tvorbu tiskových sestav (neboli reportů) z dat OR-SYSTEMu s využitím frameworku Hibernate. OR-SYSTEM je základním informačním systémem pro obchodní a výrobní společnosti, který vyvíjí právě sama společnost OR-CZ, spol. s r.o.

V současnosti, při stále rostoucím množství shromažďování dat a informací, je velmi složité s těmito daty a informacemi nejen dále pracovat, ale také poskytnout nějaký srozumitelný výstup usnadňující jejich interpretaci člověkem.

Pokud se například obchodníkům ozve jeden z jejich nedočkavých zákazníků a bude chtít poslat e-mailem informaci, v jakém stavu se právě nachází jeho zakázka, jak mu má obchodník v dané chvíli vyjít vstříc, když sám hned odpověď nezná? Jak nejefektivněji a nejrychleji předat zákazníkovi to, o co si žádá? V jakém formátu mu nalezené informace zaslat? Nebo jak zákazníkům z obrovského množství dat vytvořit přehlednou fakturu, aby se v ní kdokoli vyznal a exportovat ji do formátu, ve kterém fakturu požadují zaslat? Co když ji někteří zákazníci ale potřebují v tištěné podobě? Ovšem nejde jen o to, podat souhrnné informace zákazníkům, ale také si je předávat v přehledné formě i v rámci podniku, kde je potřeba výsledky zaznamenávat a průběžně kontrolovat.

Jedním z řešení přehledných výstupů jsou tiskové sestavy vytvořené pomocí vhodného reportovacího nástroje. Otázkou zůstává, jaký reportovací nástroj je pro podnik právě ten nejvhodnější. To samozřejmě záleží na mnoha ohledech dané firmy a na požadavcích jejích zákazníků, jelikož v dnešní době na trhu existuje spousta generátorů tiskových sestav, ze kterých si lze vybírat.

V této práci se zabývám jedním z těchto generátorů, a to JasperReports. Proč byl vybrán právě tento? Důvod byl jednoduchý – JasperReports je open-source nástroj (tedy zcela

zadarmo) postavený na jazyku Java, což je v dnešní době nejvíce rostoucí programovací jazyk a je také open-source. Právě tyto důvody odpovídaly požadavkům firmy OR-CZ, která se snaží zcela přejít na platformu Java. Ale protože by si někteří zákazníci tiskové sestavy chtěli vytvářet sami, byl JasperReports vybrán i z důvodu, že je možné si k tomuto nástroji stáhnout i grafický návrhář pro usnadnění tvorby návrhů sestav. Jako grafický návrhář byl zvolen WYSIWYG editor iReport, který je také zcela zadarmo a napsán v jazyce Java.

1. Objektově relační mapování pomocí nástroje Hibernate

1.1. Objektově relační mapování

U většiny programovacích jazyků v současnosti převládá objektově orientovaný přístup, přičemž k nejrozšířenějším objektově orientovaným programovacím jazykům patří Java. Ale u databází se objektově orientovaný přístup tolik neujal, v současné době stále převládají rozšířené relační databáze. Jelikož se jedná o dva odlišné přístupy, je potřeba odstínit rozdíly mezi nimi, což je úlohou objektově relačního mapování. K nejrozšířenějším nástrojům umožňující ORM patří framework Hibernate.

Framework je rámec sloužící k usnadnění práce při programování aplikací. Díky frameworkům nemusí vývojáři a programátoři psát opakující se kódy stále dokola a mohou se tak zabývat podstatnějšími věcmi. Framework může zahrnovat knihovny, pomocné programy a osvědčené postupy pomocí kterých zajišťuje přehlednost.

1.2. Java jako objektově orientovaný programovací jazyk

Java je jedním z nejpoužívanějších programovacích jazyků. Díky své hardwarové nezávislosti může být použita na čipových kartách, mobilních telefonech, aplikacích pro desktopové počítače…. Kromě toho, že je tento jazyk nezávislý na hardwaru, vyznačuje se také nezávislostí na operačním systému.

Jedny ze základních vlastností objektově orientovaných jazyků jsou zapouzdření, dědičnost a polymorfismus, které budou popsány v následující kapitole.

1.2.1. Objekty a třídy

Objekty v programování mají stejně jako reálné objekty své stavy (v proměnných) a chování (v metodách). Objekt je instancí nějaké třídy. Třída (objektový typ) popisuje třídy objektů, které se vyskytují v řešeném problému, je to datový typ, který si může definovat

sám programátor. Každý objekt (instance třídy) obsahuje atributy, které popisují vlastnosti daného objektu.

K zacházení s jednotlivými objekty se používají operace, které jsou součástí definice třídy a objekty je potom volají, což zlepšuje přehlednost programu. Tyto operace, které se nazývají metodami, *„pracují s vnitřním stavem objektu a slouží jako primární mechanismus pro komunikaci mezi objekty. Pokud objekt skrývá svůj vnitřní stav a požaduje, aby všechny interakce probíhaly pomocí jeho metod, označuje se to jako zapouzdření dat (data encapsulation).“* [1, s. 51]

Pokud se při definici nové třídy použije jako datová složka (atribut) již existující třída, tak se hovoří o skládání (kompozici) objektů.

Další ze základních vlastností objektově orientovaných jazyků je dědičnost. Díky dědičnosti lze od existující třídy odvodit novou třídu a lze k ní přidat nové atributy či metody. Nová třída, která je odvozená od existující, se nazývá potomek a dědí všechny atributy a metody předka, což je již existující třída (u potomka nelze odstranit nějaký atribut či metodu předka). Pokud třídy mají společné vlastnosti, je tedy možné tyto vlastnosti zahrnout v jedné třídě (předkovi), které pak ostatní třídy budou od tohoto předka dědit. Pomocí dědičnosti se vytváří hierarchická struktura daných tříd.

Další vlastností objektově orientovaného programování je polymorfismus. U objektově orientovaného programování platí, že potomek může zastoupit předka, což znamená, že tam, kde v programu očekáváme instanci předka, můžeme použít instanci potomka. Nemusíme se tak starat o skutečný typ instance (všechny typy instance obsahují tutéž metodu, protože ji obsahuje jejich předek).

1.3. Relační databáze

Data jsou v relační databázi uložena v relacích (tabulkách) představujících subjekty. Subjektem, který je reprezentován danou tabulkou, může být objekt nebo nějaká událost. V případě, že je subjektem objekt, tabulka představuje něco, na co si lze sáhnout (např. odběratel, výrobek, materiál…). Každý objekt má vlastnosti, které lze uložit jako data. Pokud je subjektem událost, potom tabulka představuje něco, k čemu dochází v jistém

okamžiku a má vlastnosti, které je vhodné zaznamenat. Stejně jako vlastnosti objektů lze i vlastnosti událostí uložit jako data (mohou to být například výsledky nějakých testů). [2]

„Tabulky lze definovat sloupci a příslušným názvem. Data jsou ukládána jako řádky tabulky. Mezi tabulkami je možné vytvořit vzájemné vazby.“ [3, s. 4] Sloupce tabulky definují atributy, neboli pole reprezentující vlastnosti subjektů, kde každá tabulka má nejméně jeden atribut, a to primární klíč. Řádky tabulky obsahují záznamy, které tvoří množina atributů, kde každý záznam v databázi je identifikován jedinečnou hodnotou (primárním klíčem).

Primární klíč je tedy atribut, popř. minimální množina atributů, které jednoznačně identifikují záznam v tabulce. Je – li primární klíč množina atributů, hovoří se o složeném primárním klíči. Primární klíč může být v dané tabulce pouze jeden. Slovo „jednoznačně“ říká, že data ve sloupci, nad kterým je vytvořen primární klíč, se nesmí opakovat a nesmí mít hodnotu „`null`“. [4] Z toho také vyplývá, že primární klíč by měla obsahovat každá tabulka, jelikož zajišťuje integritu dané tabulky a napomáhá při vytváření vztahů s ostatními tabulkami. [2]

Kromě primárních klíčů napomáhají při vytváření vztahů mezi tabulkami i cizí klíče. Cizí klíče, stejně jako primární, mají také zajišťovat integritu, ale tentokrát ne integritu dané tabulky, ale integritu vztahů. Hodnoty cizích klíčů pak musí vycházet z hodnot primárních klíčů, jelikož tím bude zajištěno, že záznamy v daných tabulkách spolu budou pokaždé náležitě souviset. [2] Na rozdíl od primárních klíčů cizí klíč nemusí být pouze jeden, ale v tabulce jich může být obsaženo i více.

1.3.1. Vztahy

Vztah mezi dvěma tabulkami existuje, pokud záznam jedné tabulky lze odkázat na záznam jiné. Mezi tabulkami se rozlišují tyto vztahy: one-to-one (1:1), one-to-many (1:N), many-to-one (N:1) a many-to-many (M:N).

1.3.1.1. Vztah one-to-one

V tomto vztahu platí, že jeden záznam v tabulce odpovídá pouze jednomu záznamu z druhé tabulky a naopak. Je tedy velmi pravděpodobné, že cizí klíč druhé tabulky bude také primárním klíčem. Vztah 1:1 se příliš často nepoužívá. Příkladem tohoto vztahu může být, že v jednom manželství v České republice může mít manžel pouze jednu manželku a manželka může mít pouze jednoho manžela.

1.3.1.2. Vztah one-to-many (many-to-one)

Vztah one-to-many se vyskytuje mezi tabulkami, když jeden záznam v jedné tabulce (primární klíč) odpovídá více záznamům z tabulky druhé (cizí klíč), kde ale jeden záznam v této druhé tabulce odpovídá jen jednomu záznamu první tabulky. Např.: k odběrateli můžeme přiřadit několik zakázek, ale určitou zakázku je možné spojit pouze s jedním odběratelem.

1.3.1.3. Vztah many-to-many

Jestliže by se porušila u vztahu one-to-one ta část, kde jeden záznam v druhé tabulce odpovídá jen jednomu záznamu z druhé tabulky, vyjde z toho vztah many-to-many. To znamená, že vztah many-to-many se objevuje u tabulek, kde jeden záznam z jedné tabulky (primární klíč) odpovídá více záznamům z druhé tabulky (cizí klíč) a jeden záznam z této druhé tabulky může odpovídat i více záznamům z tabulky první (v tomto případě se hovoří o složeném primárním klíči). Známým příkladem vztahu M:N je, že autor může napsat více knih a kniha může být napsána více autory.

1.4. Hibernate

Nástroj Hibernate byl založen roku 2001 Gavinem Kingem. Později podporovala projekt firma JBoss, ke které Gavin King přešel. Hibernate implementuje Java Persistence API 2.0

od roku 2010, kdy byla uvolněna verze 3. V současné době je nejnovější verzí Hibernate 4.1.0 (únor 2012). [5]

JPA (Java Persistence API) je standard zajišťující persistenci dat (možnost uchovávání dat i po skončení aplikačního programu) v relačních databázích. Aplikace, které používají rozhraní JPA jsou absolutně nezávislé na použitém frameworku persistence. Existují další projekty a frameworky poskytující implementaci JPA (TopLink, OpenJPA). [6]

Jak již bylo řečeno, Hibernate je jeden z nejznámějších a nejrozšířenějších nástrojů, který umožňuje objektově relační mapování a je napsán v programovacím jazyce Java. Tento nástroj také řeší problémy persistence dat. Hibernate je licencován pod LGPL (Lesser General Public License), což dovoluje použití knihovny ve vlastním softwaru, pokud je tato knihovna k softwaru pouze linkovaná. Tento software je možné šířit pod libovolnými licenčními podmínkami a může to tedy být jak svobodný tak proprietární software. [7] To podle různých výkladů, jak jsem pochopila, znamená, že se mohou provádět volání funkcí v knihovně nebo vytvářet odvozené třídy od tříd v knihovně. Není už ale možné zkopírovat část zdrojového kódu a použít ve vlastním zdrojovém kódu.

Tento nástroj umožňuje vytvářet dotazy na data svým vlastním jazykem Hibernate Query Language (HQL), který je obdobou jazyka SQL.

Hlavním úkolem toho nástroje je mapovat Java třídy z databázových tabulek a naopak. K mapování se používají tzv. mapovací soubory, což jsou běžné XML soubory, nebo anotace. V této práci se budu zabývat mapovacími soubory.

1.4.1. Konfigurace Hibernate

Základní konfiguraci Hibernate obsahuje XML soubor hibernate.cfg.xml, ve kterém je definováno JDBC připojení k databázi (IP adresa databázového serveru, port, jméno služby, přístupové údaje k databázovému schématu) a seznam všech mapovacích souborů hbm.xml. Tato konfigurace je znázorněna v ukázce kódu – *Soubor hibernate.cfg.xml*. Druhá možnost, jak nastavit přístup k databázi, je pomocí souboru hibernate.properties (*Ukázka kódu 2 – Soubor hibernate.properties*)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
      oracle.jdbc.OracleDriver</property>
    <property name="hibernate.connection.url">
      jdbc:oracle:thin:@//localhost:port/DBname</property>
     <property name="hibernate.connection.username">
      username</property>
    <property name="hibernate.connection.password">
      passwd</property>
    <mapping resource="cz/orcz/ors/Aka.hbm.xml"/>
    <mapping resource="cz/orcz/ors/Az.hbm.xml"/>
    <mapping resource="cz/orcz/ors/Dka.hbm.xml"/>
     …
  </session-factory>
</hibernate-configuration>
```

Ukázka kódu 1 – Soubor hibernate.cfg.xml

```
#JDBC driver
hibernate.connection.driver_class=oracle.jdbc.driver.OracleDriver
#JDBC URL
hibernate.connection.url=jdbc:oracle:thin:@localhost:port:DBname
#uzivatel
hibernate.connection.username=username
#heslo uživatele
hibernate.connection.password=passwd
#vypíše SQL příkazy do konzole
hibernate.show_sql=true
```

Ukázka kódu 2 – Soubor hibernate.properties

1.4.2. Mapovací soubory

K mapování databázových tabulek na Java třídy a naopak slouží tzv. mapovací soubory, které definují vztahy mezi různými objekty. Pro každou třídu existuje jeden mapovací XML soubor hbm.xml. Příklad mapovacího souboru pro tabulku ODP (sdílená cesta dokumentu) je obsažen v ukázce kódu – *Mapovací soubor* (této tabulce bude odpovídat Java třída Odp, která bude v balíku cz.orcz.ors).

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 20.12.2011 10:34:33 by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
  #název třídy, databázové tabulky, databázového schematu
```

```
  <class name="cz.orcz.ors.Odp" table="ODP" schema="ordbors">
    #název primárního klíče odpKn (systémové číslo), datový typ
    <id name="odpKn" type="int">
      <column name="ODP_KN" precision="9" scale="0" />
      <generator class="assigned" />
    </id>
    #mapování vazby one-to-one (vazba odpKn na oddKn (odd = dokumenty
    objektů, oddKn = jednoznačná identifikace dokumentu))
    <one-to-one name="odd" class="cz.orcz.ors.Odd" update="false"
    insert="false" fetch="select">
      <column name="ODP_KN" precision="9" scale="0"
      not-null="true" unique="true" />
    </one-to-one>
    #název atributu (odpPath = sdílená cesta k dokumentu OR-SYSTEMu),
    datový typ
    <property name="odpPath" type="string">
       <column name="ODP_PATH" length="200" />
    </property>
  </class>
</hibernate-mapping>
```

Ukázka kódu 3 – Mapovací soubor

1.4.3. Java class

Nyní se tedy mohou vytvořit Java třídy mapovacím souborům odpovídající. V ukázce kódu – *Java class* je znázorněna odpovídající třída k předchozímu příkladu - `Odp`, která má tři atributy: primární klíč `OdpKn`, odkaz na další tabulku (`Odd`) a atribut `OdpPath`.

```
package cz.orcz.ors;

public class Odp  implements java.io.Serializable {
     private int odpKn;
     private Odd odd;
     private String odpPath;

    public Odp() {
    }
    public Odp(int odpKn, Odd odd) {
        this.odpKn = odpKn;
        this.odd = odd;
    }
    public Odp(int odpKn, Odd odd, String odpPath) {
       this.odpKn = odpKn;
       this.odd = odd;
       this.odpPath = odpPath;
    }
    public int getOdpKn() {
        return this.odpKn;
    }
    public void setOdpKn(int odpKn) {
        this.odpKn = odpKn;
```

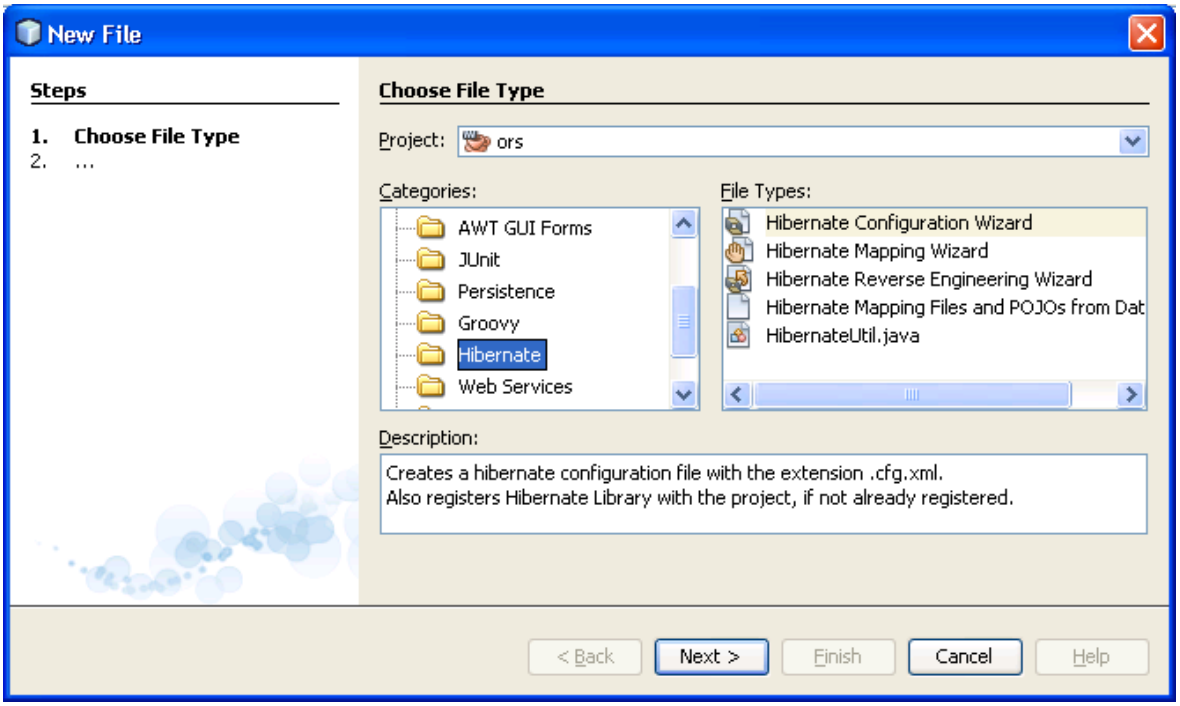
```
    }
    public Odd getOdd() {
        return this.odd;
    }
    public void setOdd(Odd odd) {
        this.odd = odd;
    }
    public String getOdpPath() {
        return this.odpPath;
    }
    public void setOdpPath(String odpPath) {
        this.odpPath = odpPath;
    }
}
```

Ukázka kódu 4 – Java class

Tyto třídy lze namapovat ručně, nebo je vygenerovat například v prostředí NetBeans automaticky. V následující kapitole bude popsáno, jak se Java třídy z existujícího datového modelu pomocí nástroje Hibernate v tomto vývojovém prostředí tvoří.

1.4.4. Použití NetBeans – generování Java tříd z databázových tabulek

Nejprve je potřeba nastavit přístup k databázi (vytvořit tedy připojení ke konkrétní databázi, s kterou se bude pracovat). Poté se vytvoří nový projekt, pomocí kterého se vygeneruje konfigurační soubor hibernate.cfg.xml (Hibernate Configuration Wizard), mapovací soubory .hbm.xml (Hibernate Mapping Wizard) a nakonec samotné Java třídy Hibernate Mapping Files and POJOs from Database).



Obrázek 1 – Hibernate v prostředí NetBeans
Zdroj: Printscreen - NetBeans

2. JasperReports

2.1. Historie

Zakladatelem JasperReports je Teodor Danciu. Pro projekt, na kterém Teodor Danciu pracoval, potřeboval použít nějaký reportovací nástroj. Protože se mu však současná řešení těchto nástrojů zdála příliš nákladná, rozhodl se vytvořit svůj vlastní. Na JasperReports začal pracovat v červnu roku 2001, v září 2001 registroval projekt na Sourceforge.net. Verze 0.1.5 byla uveřejněna už v listopadu 2001. Od tohoto okamžiku se JasperReports stal velmi populárním. Od roku 2005 jeho vývoj sponzoruje firma JasperSoft, která také poskytuje podporu a servis pro JasperReports. [8]

2.2. Co je JasperReports

JasperReports není nic jiného než open-source Java knihovna (je tedy zcela zdarma), která usnadňuje přidat reportovací schopnosti do Java aplikací. JasperReports byl tedy vyvinut jako nástroj pro tvorbu tiskových sestav neboli reportů. Tento prostředek pro generování sestav je napsán v Javě (je tedy přenositelný napříč platformami (Linux, Windows)) a pracuje s XML soubory. Jelikož JasperReports není samostatný nástroj, není určen pro koncové uživatele.

JasperReports je stejně jako Hibernate licencován pod LGPL licencí.

2.3. Možnosti JasperReports

Určitě stojí za zmínku, jaké má JasperReports možnosti. S nadsázkou řečeno, jak jednoduše a hravě se s pomocí JasperReports dají vytvářet reporty.

JasperReports umožňuje generovat reporty obsahující tabulky, obrázky, různé typy grafů (koláčový, sloupcový, spojnicový, …), čárových kódů (EAN128, Code39, 2of5, …), grafických elementů, jako jsou čáry, čtverce, kružnice atd.

Tiskové sestavy mají poskytnout přehledný výstup z velkého množství dat pro snadnější interpretaci informací. JasperReports podporuje několik datových zdrojů pomocí speciálního rozhraní `JRDataSource`. [9] Jako datový zdroj lze použít JDBC, Hibernate nebo například jednoduchý XML či CSV soubor. Dotazovat na data se tedy dá například pomocí jazyka SQL (Structured Query Language) nebo HQL (Hibernate Query Language).

Reporty generované pomocí JasperReports mohou být exportovány do několika formátů. Například je možné uložit výstup ve formátu XML, ODT, CSV, HTML, RTF, PDF, XLS,… Při exportu do PDF je třeba si dávat pozor na kódování. Pokud nebude nastaveno správně, některá diakritická znaménka by se mohla zobrazovat špatně. Proto by u každého pole mělo být nastaveno kódování `pdfEncoding="Cp1250"`.

2.3.1. Životní cyklus reportu

Obrázek 2 – Tvorba tiskové sestavy
Zdroj [8]

JasperReports vytváří výstup podle vstupní šablony – XML souboru s příponou JRXML. Tento XML soubor říká, jak výsledná sestava bude vypadat, přičemž je možné jej napsat ručně, nebo vytvořit pomocí grafického návrháře tiskových sestav. [10] Kompilací JRXML šablony vznikne binární soubor JASPER. Pokud by uživatel neměl jakýmkoliv způsobem do šablony zasahovat, lze distribuovat právě tento soubor. Pro vygenerování konečné sestavy je použita již zkompilovaná šablona JRXML – soubor JASPER, který se naplní daty. Poté se uživatel může rozhodnout, jestli sestavu pošle na tiskárnu, nebo ji uloží do některého z možných formátů.

Naplněné soubory mohou být uloženy do JasperPrint souboru, který má koncovku JRPRINT. Tyto soubory se dají prohlédnout v některém speciálním JasperPrint prohlížeči nebo exportovat do určitých formátů.

2.3.2. Rozvržení tiskových sestav

JasperReports dává možnost roztřídit data do několika sekcí, a to:

- **Titul stránky (Title)**, který se zobrazuje pouze na úplném začátku reportu, tedy na první stránce.
- **Hlavička stránky (Page Header)**, ta je zobrazena na začátku každé stránky.
- **Hlavička** nejdůležitější **sekce Detail (Column Header)** se zobrazuje před sekcí Detail na každé stránce (pokud není použita sekce Group).
- **Hlavička skupiny (Group Header) - s**ekce Group umožňují seskupovat data ze sekce Detail podle potřeby (např.: seskupit zákazníky podle měst), přičemž pokud je tato sekce použita, zobrazuje se před sekcí Detail.
- Poté následuje zmiňovaná sekce **Detail**, hlavní obsah stránky. Zobrazuje údaje z datového zdroje.
- Sekce **Patička skupiny (Group Footer)** ukončuje skupinu, je tedy zobrazena za sekcí Detail.
- **Patička Detail (Column Footer)** je zobrazena na konci každé stránce sekce Detail (pokud není použíta sekce Group).
- **Patička stránky (Page Footer)** se zobrazuje na konci každé stránky.

- **Patička poslední stránky (Last Page Footer)** se zobrazuje na konci poslední strany. Tato sekce se zobrazuje na úkor sekce Page Footer (pokud tedy bude použita sekce Last Page Footer, potom se sekce Page Footer na poslední straně nezobrazí).
- **Přehled (Summary)**, tato sekce obsahuje závěrečná shrnutí a součty.
- **No data** – pokud datový zdroj nebude obsahovat žádná data, sestava bude zobrazovat text, který je obsažen v této sekci (dalo by se říci, že tato sekce slouží jako kontrola správného nastavení datového zdroje).
- Sekce **Pozadí (Background)** definuje pozadí pro každou stránku v sestavě.

Samozřejmě není nutností použít všechny tyto sekce. Pokud například nebude potřeba hlavička stránky, je možné ji vynechat. Tisk všech sekcí lze podmínit určením výrazu `<printWhenExpression>`.

2.3.3. Datová pole

Datová pole jsou pole vybraná vlastním dotazem (*Ukázka kódu 5 – Dotazování*). Pole je definováno svým jménem a datovým typem. V sestavě se označují jako `$F{NázevPole}`. Vlastní definice pole je pak znázorněna v ukázce kódu – *Definice pole*, kde atribut `dkaNr` představuje číslo zákazníka datového typu long.

```
<queryString language="hql">
      <![CDATA[FROM Dka WHERE dkaNr == 1]]>
</queryString>
```

Ukázka kódu 5 – Dotazování

```
<field name="dkaNr" class="java.lang.Long"/>
```

Ukázka kódu 6 – Definice pole

2.3.4. Výrazy

Kromě datových polí lze do sestavy vkládat výrazy, ve kterých je možné použít již zmiňovaná datová pole, parametry, proměnné (viz dále) nebo také například Java třídy. Výrazy slouží k vyhodnocování náročnějších operací, vzorců… Základní jazyk pro psaní

výrazů je Java, ale lze také použít JavaScript nebo Groovy. Použití některých výrazů lze vidět v ukázce kódu – *Výrazy*.

```
<textFieldExpression>
      <![CDATA["Termín: " + $P{01} + " - " + $P{02}]]>
</textFieldExpression>

<printWhenExpression>
      <![CDATA[$F{pocet}.intValue() > 0]]>
</printWhenExpression>

<imageExpression>
      <![CDATA["images//logo.png"]]>
</imageExpression>
```

Ukázka kódu 7 – Výrazy

2.3.5. Parametry

Parametry se označují jako `$P{název}`. Existuje sada systémových parametrů a pak také uživatelské parametry. Příklady systémových:

- REPORT_PARAMETERS_MAP – obsahuje systémové a uživatelské parametry v sestavě
- REPORT_DATA_SOURCE – obsahuje odkaz na instance třídy `JRDataSource` obsahující zdroj dat
- REPORT_CONNECTION – obsahuje odkaz na `java.sql.Connection` předaný do reportu pro připojení k databázi
- REPORT_LOCALE – určuje jazyk pro generování sestav, které jsou následně dle lokalizace přeloženy do konkrétního jazyka
- REPORT_RESOURCE_BUNDLE – instance třídy `java.util.ResourceBundle`, pro lokalizaci reportu
- REPORT_SCRIPTLET – instance třídy `JRAbstractScriptlet` obsahující odkaz na Scriptlet vytvořený uživatelem (více viz kapitola *2.3.12 Scriptlety*)
- REPORT_MAX_COUNT – umožňuje omezit počet záznamů zpracovávaných v sestavě
- IS_IGNORE_PAGINATION – pro potlačení stránkování sestavy (sestava bude jedna dlouhá stránka)

- HIBERNATE_SESSION – parametr, který se předává při vkládání subreportů do hlavního reportu

Uživatelské parametry se používají pro parametrizaci dotazu do databáze. Je-li například potřeba vygenerovat report pro konkrétního zákazníka nebo pro určitý rozsah zákazníků, použijí se k tomu parametry. V ukázce kódu – *Parametry* jsou deklarovány parametry pro rozsah zákazníků pro tisk s názvem 01, 02 typu Integer a s jejich defaultně nastavenými hodnotami. V ukázce kódu – *Použití parametrů* je znázorněno následné použití těchto parametrů, kde atribut `dkaNr` představuje číslo zákazníka.

```
<parameter name="01" class="java.lang.Integer">
      <defaultValueExpression>
            <![CDATA[1]]>
      </defaultValueExpression>
</parameter>
<parameter name="02" class="java.lang.Integer">
      <defaultValueExpression>
            <![CDATA[999999999]]>
      </defaultValueExpression>
</parameter>
```

Ukázka kódu 8 – Parametry

```
from Dka
where dkaNr>=$P{01} and dkaNr<=$P{02}
```

Ukázka kódu 9 – Použití parametrů

2.3.6. Proměnné

Proměnné se označují jako `$V{název}` a používají se pro výpočty prováděné přímo v sestavě. Jako u parametrů i zde existuje sada systémových proměnných:

- PAGE_NUMBER – číslo konkrétní stránky
- PAGE_COUNT - počet stran obsažených celkově v dokumentu
- COLUMN_NUMBER - aktuální číslo sloupce v případě vícesloupcové sestavy
- COLUMN_COUNT - počet sloupců
- REPORT_COUNT - pořadové číslo zpracovávaného záznamu

Uživatelské proměnné se dají upotřebit pro vlastní výpočty mezisoučtů, součtů…. Například proměnná „A“ v ukázce kódu – *Proměnná* určí počet zakázek, které byly

splněny v termínu (pokud se pole PLNENI bude rovnat „ANO“, tak se zakázka započítá, jinak ne).

```
<variable name="A" class="java.lang.Integer" calculation="Count">
      <variableExpression>
            <![CDATA[$F{PLNENI}.equals("ANO")?$F{AZ_ANR}:null]]>
      </variableExpression>
</variable>
```

Ukázka kódu 10 – Proměnná

2.3.7. Styly

Je možné, že se v sestavě několikrát bude opakovat tentýž styl, například stejný rámeček u více polí, nebo se v sekci Detail bude střídat barva řádků (každý druhý řádek bude šedý). Tyto požadavky lze řešit pomocí uživatelem vytvořeného stylu, který se použije u každého pole, které se tímto stylem označí. V ukázce kódu – *Styl* se definuje styl s názvem „ramecek“, který se vyznačuje pouze černým ohraničením s tloušťkou pera 0,5. Tento styl se poté použije v části <reportElement> pomocí style="ramecek".

```
<style name="ramecek">
      <box      topPadding="0"      leftPadding="0"      bottomPadding="0"
      rightPadding="0">
            <pen lineWidth="0.5"/>
            <topPen lineWidth="0.5"/>
            <leftPen lineWidth="0.5"/>
            <bottomPen lineWidth="0.5"/>
            <rightPen lineWidth="0.5"/>
      </box>
</style>
```

Ukázka kódu 11 – Styl

2.3.7.1. Podmíněný styl

Podmíněný styl <conditionalStyle> je součástí základního stylu a používá se, jestliže se tento styl má aplikovat na základě splnění zadané podmínky. Příkladem může být, že každý druhý řádek bude mít šedé pozadí. Definice podmíněného stylu a jeho následné použití u rámečku <frame> je znázorněno v ukázce kódu – *Podmíněný styl*.

```
<style name="radky">
      <conditionalStyle>
```

```
            <conditionExpression>
                  <![CDATA[$V{REPORT_COUNT}.intValue()%2 == 0]]>
            </conditionExpression>
            <style mode="Transparent" backcolor="#999999"/>
      </conditionalStyle>
</style>

<frame>
      <reportElement style="style1" x="0" y="0" width="276" height="25"/>
      ...
</frame>
```

Ukázka kódu 12 – Podmíněný styl

2.3.8. Seskupování

Skupina (Group) představuje způsob, jak v sestavě seskupovat data. K tomu slouží sekce Group Header a Group Footer. Poněvadž počet skupin v sestavě není omezen, může jich uživatel vložit tolik, kolik potřebuje. V případě použití více skupin by se dalo hovořit o skupinách a podskupinách. Pro každou z těchto skupin a podskupin je pak možné vytvářet například různé mezisoučty prostřednictvím proměnných. U jednotlivých skupin lze pomocí `<printWhenExpression>` nastavit, za jakých podmínek se mají tisknout. Než se data začnou seskupovat (než je požita sekce Group), musí být nejdříve setříděna - nesmí tedy v HQL (popř. SQL) dotazu chybět příkaz `order by`. V ukázce kódu – *Skupina* je znázorněna skupina s názvem „dodavatel“, která je seřazena podle krátkého jména dodavatele DKA_KURZ (uvedeno v elementu `<groupExpression>`).

```
<group name="dodavatel">
      <groupExpression>
            <![CDATA[$F{DKA_KURZ}]]>
      </groupExpression>
      <groupHeader>
      </groupHeader>
      <groupFooter>
      </groupFooter>
</group>
```

Ukázka kódu 13 – Skupina

2.3.9. Grafy

JasperReports umožňuje do sestav vkládat hned několik typů grafů (koláčový, sloupcový, plošný, spojnicový, bodový, 3D grafy a spoustu dalších). Informace o grafu se vyskytují v elementu `<chart>`, který patří do elementu odpovídajícímu typu grafu (např. `<pieChart>` pro koláčový graf, který je použit v ukázce kódu – *Koláčový graf* a znázorňuje platební morálku zákazníků).

```
<pieChart>
  <chart renderType="draw" theme="generic">
    <reportElement mode="Transparent" x="165" y="0" width="390"
    height="246"/>
  </chart>
  <pieDataset maxCount="30" minPercentage="1.0">
    <keyExpression><![CDATA[$F{splatnost}]]></keyExpression>
    <valueExpression><![CDATA[$F{hodnota}]]></valueExpression>
    <labelExpression>
      <![CDATA[$F{splatnost}+"dnů po splatnosti - "+$F{hodnota}+" CZK"]]>
    </labelExpression>
  </pieDataset>
  <piePlot>
    <plot/>
  </piePlot>
</pieChart>
```

Ukázka kódu 14 – Koláčový graf

Element `<keyExpression>` je identifikátor pro každý díl koláče a `<valueExpression>` je číselnou hodnotu, která definuje velikost každého dílu, reprezentuje tedy hodnotu dílů. Pomocí `<labelExpression>` se definuje text, který se bude zobrazovat u jednotlivých částí koláče.

2.3.10. Vícejazyčné reporty

Díky podpoře vícejazyčných reportů, je možné použít jednu sestavu pro různé jazyky. Díky třídě `ResourceBundle` lze text snadno lokalizovat a přeložit ho do různých jazyků. Lokalizační objekt je možné předat pomocí `resourceBundle` (např. `resourceBundle="i18n"`) v hlavním elementu `<jasperReport>`. JasperReports se pak pokusí načíst properties soubor ze stejné složky, kde je obsažen report. Nejprve se pokusí načíst soubor odpovídající zadanému názvu v `resourceBundle`, pak k názvu přidá koncovku podle lokalizace operačního systému. Nejdříve se tedy pokusí načíst

soubor i18n.properties a pak i18n_cs.properties (popř. i18n_cs_CZ.properties). Properties soubor má standardní tvar, obsahuje tedy klíč a hodnotu. Ve výrazech pak lze používat `$R{identifikator}` (*Ukázka kódu 15 – Vícejazyčný report*).

```
<textField>
  <reportElement x="0" y="0" width="500" height="20"/>
  <textElement verticalAlignment="Middle"/>
  <textFieldExpression>
    <![CDATA[$R{text}]]>
  </textFieldExpression>
</textField>
```

Ukázka kódu 15 – Vícejazyčný report

2.3.11. Subreporty

Jako další element, který lze v sestavě použít je subreport (podsestava). Jinými slovy JasperReports má schopnost do jakéhokoliv reportu vložit jiný, předem vytvořený report. Těchto podsestav samozřejmě může být v hlavní sestavě umístěno i více, jejich množství záleží na potřebě uživatele. Vytvořený subreport, který se do hlavního reportu bude vkládat, musí obsahovat parametr, pomocí kterého se subreport a hlavní report spojí (který se ze subreportu bude do hlavního reportu předávat). Jestliže bude použit jako datový zdroj Hibernate, je třeba pro správnou funkčnost sestavy předat mezi hlavním reportem a subreportem také parametr HIBERNATE_SESSION. Deklarace subreportu je znázorněna v ukázce kódu – *Subreport*.

```
<subreport>
      <reportElement x="-20" y="0" width="575" height="38"/>
      <subreportParameter name="HIBERNATE_SESSION">
            <subreportParameterExpression>
                  <![CDATA[$P{HIBERNATE_SESSION}]]>
            </subreportParameterExpression>
      </subreportParameter>
      <subreportParameter name="Anr">
            <subreportParameterExpression>
                  <![CDATA[$P{01}]]>
            </subreportParameterExpression>
      </subreportParameter>
      <subreportExpression>
            <![CDATA["jsap001_01.jasper"]]>
      </subreportExpression>
</subreport>
```

Ukázka kódu 16 – Subreport

Výraz `<subreportParameterExpression>` v elementu `<subreportParameter>` udává parametr, který se bude předávat a výraz `<subreportExpression>` určuje cestu, kde se daný subreport nachází. Jak je zde vidět, zadává se již zkompilovaná šablona JASPER.

2.3.12. Scriptlety

Pomocí Scriptletů se do sestavy můžou vkládat i samostatně vytvořené Java třídy. Scriptlet je tedy Java třída, která se používá ke spuštění složitějších operací s daty (Scriptletů lze použít i více). Scriptlet může obsahovat řadu metod, které se volají během tvorby jednotlivých částí reportu, jako je začátek stránky, skupiny nebo detailu. [9]

Scriptlet se definuje v hlavním elementu `<jasperReport>`, pomocí názvu třídy `scriptletClass="NázevTřídy"`. Pro spouštění reportů se Scriptlety z grafického návrháře iReport (kapitola *2.5 Grafický návrhář iReport*) se vytvořený projekt v Javě přidá do CLASSPATH iReportu, jinak se přidá do podpůrných knihoven programu, který generuje sestavy (viz kapitola *2.6 Běh pod Javou*). Poté se už jen použije parametr REPORT_SCRIPTLET a zavolá se metoda (*Ukázka kódu 17 – Scriptlet*).

```
<textField>
      <reportElement x="0" y="0" width="555" height="20"/>
      <textElement verticalAlignment="Middle"/>
      <textFieldExpression>
            <![CDATA[$P{REPORT_SCRIPTLET}.VytvořenáMetoda()})]]>
      </textFieldExpression>
</textField>
```

Ukázka kódu 17 – Scriptlet

2.4. Tvorba reportu

Jak již bylo řečeno, první věcí při vytváření tiskové sestavy je tvorba JRXML šablony. JRXML šablona je jednoduše XML soubor. Mohou být tedy napsány ručně, nebo se k vytvoření reportu může využít grafický návrhář.

Hlavním elementem JRXML souboru je <jasperReport>, který obsahuje spoustu dalších subelementů (viz dále, kapitola *2.4.1 Základní JRXML soubor*).

2.4.1. Základní JRXML soubor

Strukturu základního JRXML souboru představuje ukázka kódu – *Základní JRXML soubor*.

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport name="report1">
      <title>
            <band height="79"/>
      </title>
      <pageHeader>
            <band height="35"/>
      </pageHeader>
      <columnHeader>
            <band height="61"/>
      </columnHeader>
      <detail>
            <band height="125"/>
      </detail>
      <columnFooter>
            <band height="45"/>
      </columnFooter>
      <pageFooter>
            <band height="54"/>
      </pageFooter>
      <summary>
            <band height="42"/>
      </summary>
</jasperReport>
```

Ukázka kódu 18 – Základní JRXML soubor

Všechny části jsou zahrnuty v oddílu <jasperReport>. Každý prvek obsahuje část <band>, ve které jsou data zobrazující se v tiskové sestavě. Jelikož v této ukázce nebyla žádná data použita, vytvoří se pouze prázdná šablona.

2.4.2. Základní elementy

Část <detail> obsahuje pouze jeden element, a to <band>, který je jediným subelementem této části. Zato v právě zmiňované části <band> se může takových subelementů použít více. Záleží na tom, co se má v dané sestavě zobrazovat, jestli pouze

text nebo i obrázek, čárový kód, tabulka atd. *Ukázka kódu 19 – Detail* představuje pouze sestavu s textem (část <detail>).

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="report1"
pageWidth="595" pageHeight="842" columnWidth="555" leftMargin="20"
rightMargin="20" topMargin="20" bottomMargin="20">
  <queryString language="hql">
     <![CDATA[from Zakazky]]>
  </queryString>
  <detail>
    <band height="20">
      <staticText>
         <reportElement x="110" y="0" width="216" height="20"/>
         <textElement verticalAlignment="Middle">
            <font isBold="true" pdfEncoding="Cp1250"/>
         </textElement>
         <text>
            <![CDATA[Neexistuje kupní smlouva ani nabídka č.:]]>
         </text>
      </staticText>

      <textField>
         <reportElement x="326" y="0" width="101" height="20"/>
         <textElement verticalAlignment="Middle"/>
         <textFieldExpression class="java.lang.Integer">
            <![CDATA[$F{cislo_zak}]]>
         </textFieldExpression>
      </textField>
    </band>
  </detail>
</jasperReport>
```

Ukázka kódu 19 – Detail

Na začátku je možno si všimnout části <jasperReport>, ve které jsou obsaženy vlastnosti, které se týkají dané šablony, tzn. název, velikost stránky, okraje…

Jak již bylo řečeno, v kapitole *2.4.1 Základní JRXML soubor*, všechny důležité elementy, které se zobrazují ve výsledné tiskové sestavě, jsou obsaženy v části <band>.

Nyní se zaměříme na některé elementy z ukázky. Úsek <staticText> představuje jakýkoliv text, který do sestavy vepíšeme a který je nezávislý (nezávisí tedy na datech z databáze). Tento text se zapisuje do elementu <text>.

Oproti tomu v elementu `<textFiled>` mohou být obsaženy právě data z databáze (`$F{cislo_zak}`). Pole se v tomto případě zapisuje jako výraz do části `<textFieldExpression>`.

Část `<reportElement>` udává pozici, šířku a výšku jednotlivých polí. Element `<textElement>`, který může obsahovat podelement `<font>` definující údaje o konkrétním textu – zarovnání na střed a tučné písmo.

Jak je vidět, dotazy na data se zadávají ještě před částí `<detail>` v úseku `<queryString>` společně s použitým jazykem (`language="hql"`). V ukázce je použitý jazyk HQL (`![CDATA[from Zakazky]]`).

2.4.3. Další důležité elementy

V této kapitole jsou uvedeny nejvýznamnější elementy, s kterými se každý při tvorbě základního JRXML souboru pravděpodobně setká. V příloze A (*str. 87*) je obsažena výsledná sestava exportovaná do formátu PDF.

2.4.3.1. Element <queryString>

V části `<queryString>` se zadávají dotazy na data z databáze (zde pomocí jazyka Hibernate Query Language, jelikož je jako datový zdroj použit Hibernate).

```
<queryString language="hql">
   <![CDATA[FROM Dka WHERE dkaNr >= $P{01} AND dkaNr <= $P{02}]]>
</queryString>
```

Ukázka kódu 20 – Dotazy na data, HQL

2.4.3.2. Element <field>

V elementu `<field>` jsou definována data z databáze – číslo zákazníka, jméno, město a ulice.

```
<field name="dkaNr" class="java.lang.Long">
   <fieldDescription><![CDATA[ČísloZákazníka]]></fieldDescription>
</field>

<field name="dkaOrt" class="java.lang.String">
   <fieldDescription><![CDATA[Město]]></fieldDescription>
</field>

<field name="dkaName1" class="java.lang.String">
   <fieldDescription><![CDATA[Jméno]]></fieldDescription>
</field>

<field name="dkaStr" class="java.lang.String">
   <fieldDescription><![CDATA[Ulice]]></fieldDescription>
</field>
```

Ukázka kódu 21 – Pole

2.4.3.3. Sekce Title

Element `<title>` představuje titulek reportu, který se zobrazuje pouze na první straně.

```
<title>
   <band height="46">
      <frame>
         <reportElement mode="Opaque" x="0" y="0" width="555"
           height="46" backcolor="#006699"/>
          <staticText>
            <reportElement mode="Transparent" x="0" y="0"
              width="152" height="46" forecolor="#FFFFFF"/>
             <textElement verticalAlignment="Middle" markup="none">
                <font size="34" isUnderline="false"
                  isStrikeThrough="false"/>
             </textElement>
             <text><![CDATA[ADRESY]]></text>
          </staticText>
      </frame>
   </band>
</title>
```

Ukázka kódu 22 – Titulek

V ukázce kódu – *Titulek* je vidět, že nadpisem bude ta část, která je ohraničena elementem `<text>`, tedy ADRESY. Příkaz `forecolor` udává barvu písma (zde bílá barva).

2.4.3.3.1. Element <frame>

Nadpis ADRESY z předchozí kapitoly je součástí rámečku, který se označuje elementem `<frame>`. Rámeček slouží k odlišení určitého pole nebo skupiny polí, a to změnou barvy písma, barvy pozadí…. Rámečkem lze ohraničit skupinu např. Text Field nebo Static Text, které se mají zvýraznit. V tomto případě se rámeček vyznačuje modrou barvou pozadí (`backcolor="#006699"`). U rámečku je důležitá vlastnosti `mode="Opaque"`, pokud tam totiž tato vlastnost nebude, rámeček bude průhledný (v tomto případě by to znamenalo, jako kdyby modrá barva pozadí vůbec nebyla nastavena).

2.4.3.4. Sekce Page Header

Hlavička stránky je zobrazena na začátku každé stránky.

```
<pageHeader>
   <band height="21">
      <staticText>
         <reportElement x="0" y="1" width="139" height="20"
         forecolor="#006699"/>
         <textElement verticalAlignment="Middle"/>
         <text><![CDATA[Čísla a adresy zákazníků]]></text>
      </staticText>
   </band>
</pageHeader>
```

Ukázka kódu 23 – Hlavička stránky

V elementu `<pageHeader>` se bude zobrazovat obsah v části `<text>`. Tento nadpis je napsán modrou barvou (`forecolor="#006699"`) a zasazen doprostřed pole `<textElement verticalAlignment="Middle"/>`.

2.4.3.5. Sekce Column Header

Element `<columnHeader>` představující hlavičku sekce Detail se zobrazuje na každé stránce právě před touto sekcí.

```
<columnHeader>
  <band height="21">
    <frame>
      <reportElement mode="Opaque" x="0" y="1" width="437" height="20"
```

```
      backcolor="#999999"/>
      <staticText>
        <reportElement x="293" y="1" width="144" height="19"
        forecolor="#FFFFFF"/>
        <textElement verticalAlignment="Middle">
          <font isBold="true" pdfFontName="Helvetica"
          pdfEncoding="CP1250" isPdfEmbedded="true"/>
        </textElement>
        <text><![CDATA[MĚSTO]]></text>
      </staticText>
      <staticText>
        <reportElement x="107" y="0" width="186" height="20"
        forecolor="#FFFFFF"/>
        <textElement verticalAlignment="Middle">
          <font isBold="true" pdfFontName="Helvetica"
          pdfEncoding="CP1250" isPdfEmbedded="true"/>
        </textElement>
        <text><![CDATA[JMÉNO]]></text>
      </staticText>
      <staticText>
        <reportElement x="0" y="0" width="80" height="20"
        forecolor="#FFFFFF"/>
        <textElement textAlignment="Right" verticalAlignment="Middle">
          <font isBold="true" pdfFontName="Helvetica"
          pdfEncoding="CP1250" isPdfEmbedded="true"/>
        </textElement>
        <text><![CDATA[Č. ZÁKAZNÍKA]]></text>
      </staticText>
    </frame>
  </band>
</columnHeader>
```

Ukázka kódu 24 – Hlavička sekce Detail

Nadpis sekce Detail (MĚSTO, JMÉNO, ČÍSLO ZÁKAZNÍKA) bude vyobrazen tučně (`isBold="true"`) s bílým písmem (`forecolor="#FFFFFF"`) a část `<frame>` říká, že nadpisy budou na šedém pozadí (*Ukázka kódu 25 – Rámeček*).

```
<frame>
<reportElement mode="Opaque" x="1" y="0" width="554" height="20"
backcolor="#999999"/>
…
</frame>
```

Ukázka kódu 25 – Rámeček

2.4.3.6. Sekce Detail

Tato kapitola obsahuje ukázku čárového kódu `<jr:barbecue>` typu `Code128`, který se nachází v elementu `<componentElement>`. Jako čárový kód se zobrazuje číslo zákazníka (`dkaNr`), což je definováno pomocí výrazu v části `<jr:codeExpression>`.

Pro vysvětlení: `dka` = třída zákazníků, `dkaNr` = číslo zákazníka, `dkaName1` = jméno zákazníka.

```
<detail>
  <band height="70">
    <textField>
      <reportElement x="294" y="0" width="143" height="23"/>
      <textElement verticalAlignment="Middle">
        <font pdfFontName="Helvetica" pdfEncoding="CP1250"
        isPdfEmbedded="true"/>
      </textElement>
      <textFieldExpression><![CDATA[$F{dkaOrt}]]></textFieldExpression>
    </textField>
    <textField>
      <reportElement x="108" y="1" width="186" height="21"/>
      <textElement verticalAlignment="Middle">
        <font pdfFontName="Helvetica" pdfEncoding="CP1250"
        isPdfEmbedded="true"/>
      </textElement>
      <textFieldExpression><![CDATA[$F{dkaName1}]]></textFieldExpression>
    </textField>
    <componentElement>
      <reportElement x="0" y="0" width="108" height="22"/>
      <jr:barbecue
xmlns:jr="http://jasperreports.sourceforge.net/jasperreports/components"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports/co
mponents http://jasperreports.sourceforge.net/xsd/components.xsd"
type="Code128" drawText="true" checksumRequired="false">
        <jr:codeExpression>
          <![CDATA[$F{dkaNr}.intValue()]]>
        </jr:codeExpression>
      </jr:barbecue>
    </componentElement>
  </band>
</detail>
```

Ukázka kódu 26 – Detail

2.4.3.7. Sekce Page Footer

Patička stránky se zobrazuje na konci každé stránky (opak elementu `<pageHeader>`). V `<pageFooter>` mohou být například uvedeny čísla stránek nebo datum, jak je tomu v ukázce kódu – *Patička stránky*.

```
<pageFooter>
   <band height="22">
      <textField>
         <reportElement mode="Opaque" x="0" y="0" width="511"
            height="21" backcolor="#CCCCCC"/>
         <textElement textAlignment="Right"
            verticalAlignment="Middle"/>
```

```
            <textFieldExpression>
               <![CDATA["Strana "+$V{PAGE_NUMBER}+" z"]]>
            </textFieldExpression>
         </textField>
         <textField evaluationTime="Report">
            <reportElement mode="Opaque" x="511" y="0" width="44"
               height="21" backcolor="#CCCCCC"/>
            <textElement verticalAlignment="Middle"/>
            <textFieldExpression>
               <![CDATA[" " + $V{PAGE_NUMBER}]]>
            </textFieldExpression>
         </textField>
         <textField pattern="EEEEE dd MMMMM yyyy">
            <reportElement x="6" y="6" width="100" height="13"/>
            <textElement>
            <font pdfFontName="Helvetica" pdfEncoding="CP1250"
               isPdfEmbedded="true"/>
            </textElement>
            <textFieldExpression>
               <![CDATA[new java.util.Date()]]>
            </textFieldExpression>
         </textField>
      </band>
</pageFooter>
```

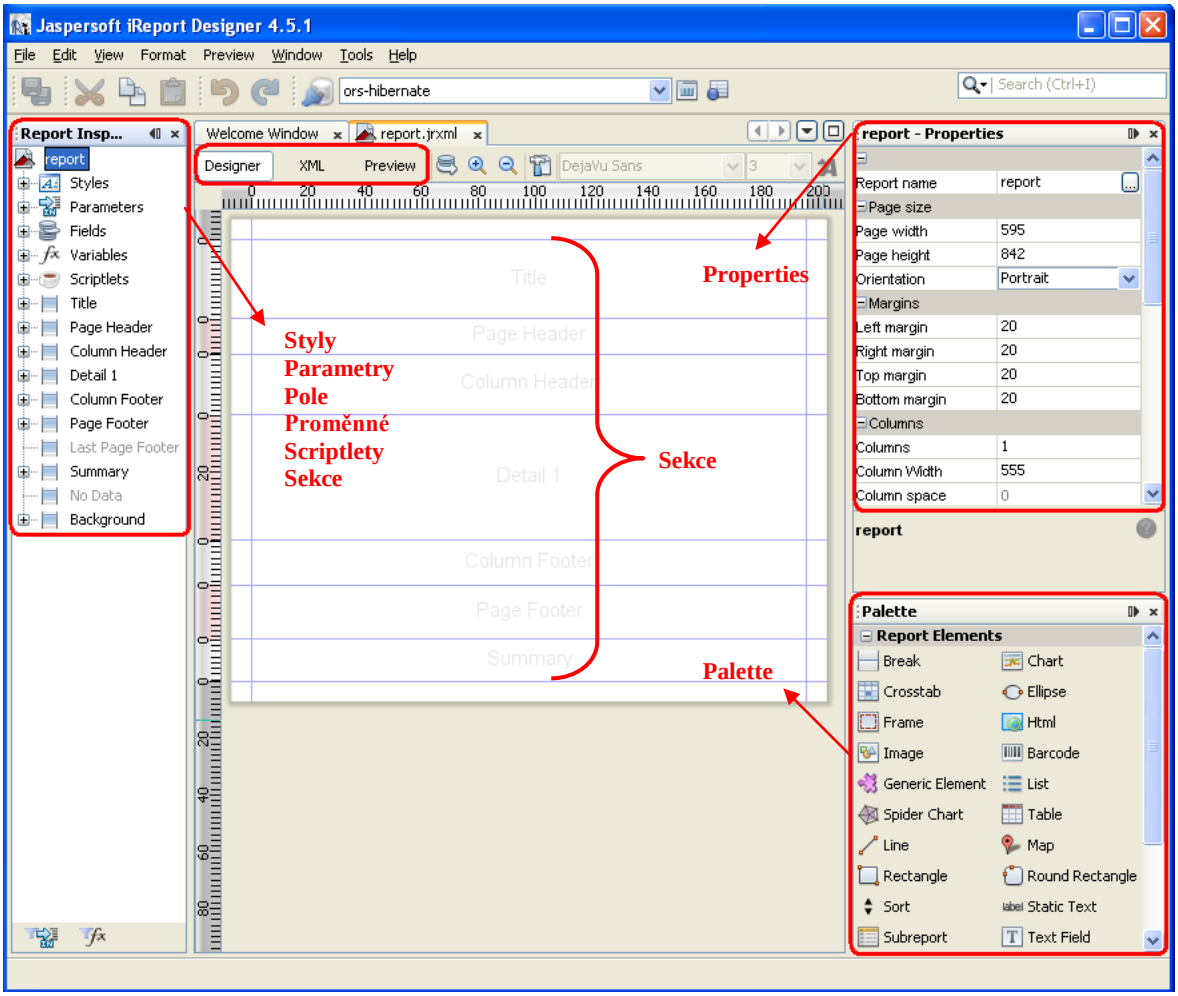
Ukázka kódu 27 – Patička stránky

2.5. Grafický návrhář iReport

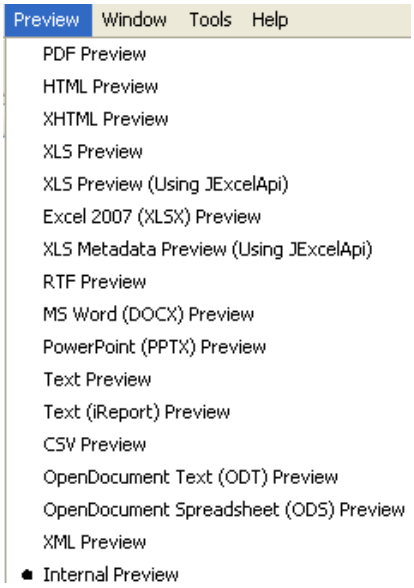
Projekt iReport byl založen v roce 2001 Guliem Toffoli.

iReport, grafický návrhář sestav pro JasperReports umožňuje vytvářet vzhled výstupu tak, jak bude vypadat při tisku (je to podobné jako v MS Wordu či Open Office Writeru). iReport tedy usnadňuje práci od ručního psaní XML šablon (JRXML souborů). V tomto grafickém návrháři lze sestavu nejen vytvořit v záložce Designer (kde jsou obsaženy sekce zmíněné v kapitole *2.3.2 Rozvržení tiskových sestav*), ale lze si šablonu také prohlédnout v XML souboru (záložka XML) a dokonce je možné si ji hotovou prohlédnout v interním prohlížeči, který tento návrhář nabízí, nebo ji exportovat do potřebného formátu.

iReport umožňuje vytvářet sestavy, JasperReports je umožňuje spouštět a vytvářet výstupy v Java aplikacích. [11]

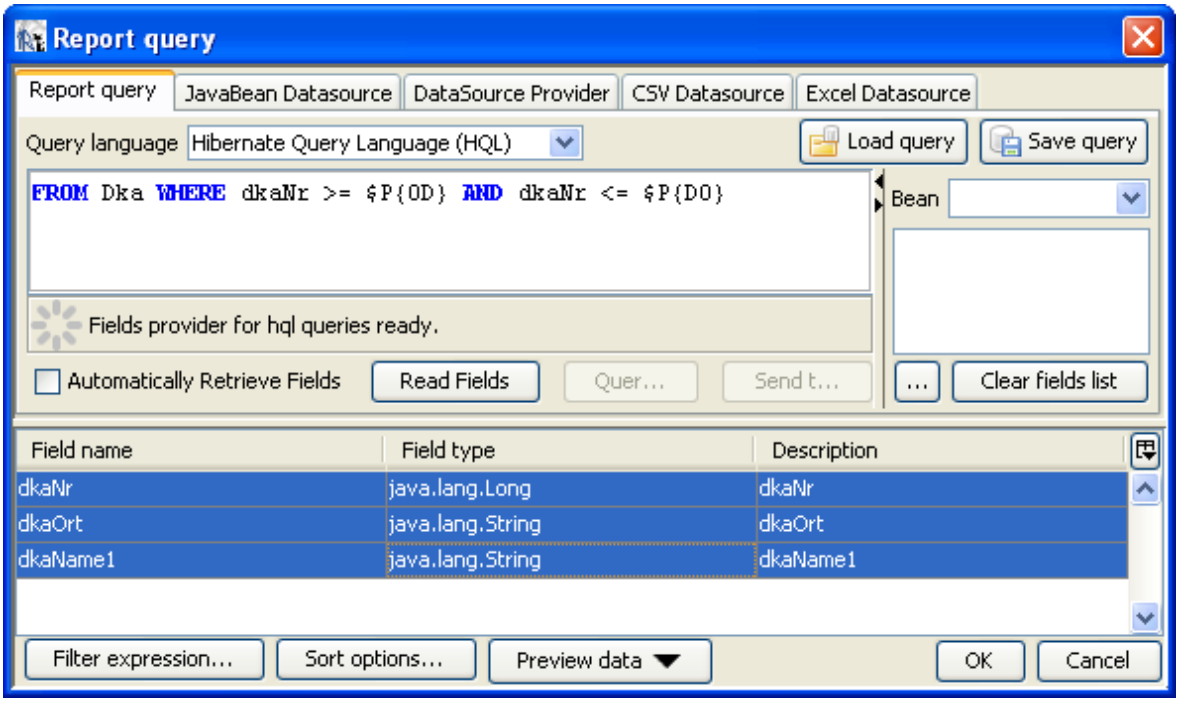


Obrázek 3 - Grafický návrhář iReport
Zdroj: Printscreen - iReport



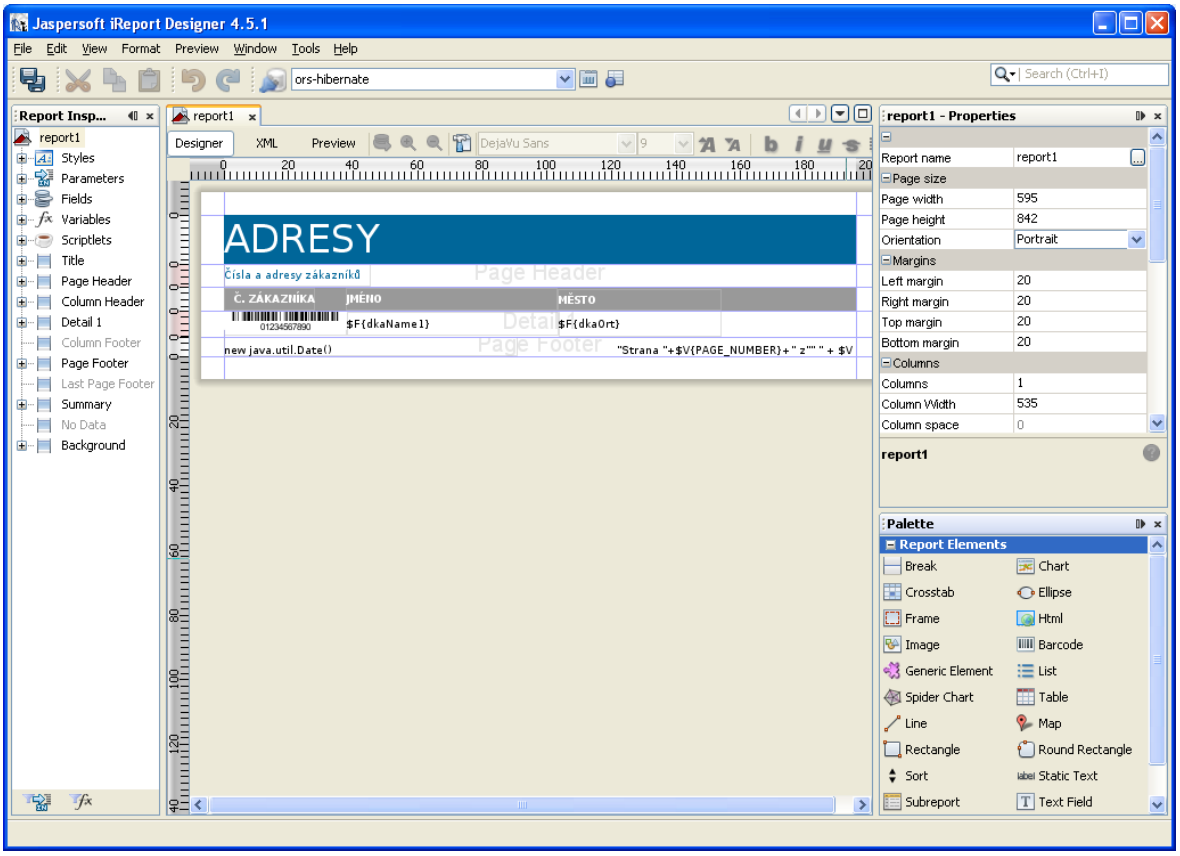
Obrázek 4 – Možnosti exportů
Zdroj: Printscreen - iReport

V kapitole *2.4 Tvorba reportu* jsme se seznámili s příkladem tvorby reportu pomocí JRXML šablony. Na obrázku – *Dotazovací okno Report Query* bude uveden totožný příklad, ale tentokrát vytvořený pomocí grafického návrháře iReport. Pole s daty se do šablony vkládají metodou „Drag and Drop“ z levé části grafického návrháře na základě HQL dotazu v tomto okně:



Obrázek 5 – Dotazovací okno Report Query
Zdroj: Printscreen - iReport

Pomocí nástroje Palette se metodou „Drag and Drop“ kromě pole s daty do šablony dají vkládat různé elementy (texty, subreporty, čárové kódy, grafické prvky, tabulky...). Prostřednictvím nástroje Properties lze nastavit různé vlastnosti jako například styl písma, velikost, font, tloušťku čar, barvy... (tyto nástroje lze vidět na obrázku – *Ukázka vytvořené šablony v iReport*).



Obrázek 6 – Ukázka vytvořené šablony v iReport
Zdroj: Printscreen – iReport

2.6. Běh pod Javou

V předchozích kapitolách bylo popsáno, jak vytvořit sestavu pomocí šablony JRXML. Tato kapitola bude obsahovat způsob, jak tuto šablonu zkompilovat, naplnit a exportovat do PDF přes příkazový řádek pomocí programu vytvořeného v Javě.

```
public class Orjasper {

    public static void main(String[] argv) throws JRException {


        Map<String, Object> params = new HashMap<String, Object>();
         JasperReport jr = JasperCompileManager.compileReport(argv[0]);

         Session session = NewHibernateUtil.getSessionFactory().getCurrentSession();
         session.beginTransaction();
          params.put("HIBERNATE_SESSION", session);
          // Hibernate session se poda jako parametr
         JasperPrint jp = JasperFillManager.fillReport(jr, params);
         session.getTransaction().commit();


         JasperExportManager.exportReportToPdfFile(jp, argv[1]);

    }
}
```

Ukázka kódu 28 – Java program, tvorba reportu

Pomocí metody `compileReport()` ze třídy `JasperCompileManager` dochází ke kompilaci šablony JRXML, kde parametrem je název této šablony i s cestou, kde je uložena, (dáno argumentem `argv[0]`. Ke generování reportu (naplnění JRXML šablony) slouží metoda `fillReport()` ze třídy `JasperFillManager`, kde prvním parametrem je zkompilovaná šablona (`jr`) a druhým parametrem jsou parametry definované v této šabloně (`params`). Metoda `exportReportToPdfFile()` ze třídy `JasperExportManager` slouží, jak sám název napovídá, k exportování naplněné šablony do formátu PDF (prvním parametrem metody je již naplněná šablona a druhým parametrem je název souboru i s cestou, kam se má sestava uložit (zde druhým argumentem `argv[1]`)).

Pokud budou vytvářeny reporty s využitím nástroje Hibernate a konfiguračního souboru hibernate.properties, bude potřeba použít `SessionFactory`, která vytvoří nebo otevře `session` pro komunikaci s databází. Nejprve se vytvoří objekt `Configuration`, který spravuje konfigurace definované v souboru hibernate.properties. Prostřednictvím objektu `Configuration` se pomocí metody `buildSessionFactory()` vytvoří `SessionFactory`. `Session` se automaticky uzavře metodou `commit()`. [6]

```
public class NewHibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Ukázka kódu 29 – Session Factory

2.6.1. Knihovny

V této podkapitole jsou popsány některé důležité knihovny, které jsou potřeba pro běh JasperReports. Tyto knihovny se musí zařadit do CLASSPATH programu.

Apache Commons je kolekcí Java knihoven zahrnující knihovny a potřebné třídy poskytující běžně používané funkce. Např. Common Digester knihovna obsahuje pomocné třídy, které slouží k inicializaci Java objektů z XML souborů.

iText je knihovna pro vytváření reportů v PDF formátech. V CLASSPATH by měla být, pokud se tiskové sestavy budou exportovat do formátu PDF nebo RTF.

Knihovna JExcelApi dovoluje pracovat s reporty ve formátu Microsoft Excel.

Kromě výše uvedených knihoven existuje i spousta dalších podpůrných knihoven, které by při použití některých prvků měly být rozhodně do programu přidány. Příkladem můžou být knihovny JFreeChart a jasperreports-chart-themes pro podporu tisku grafů, knihovna barbecue, která je efektivní při využití čárových kódů v sestavě nebo knihovna groovy-all podporující programovací jazyk Groovy, který je obdobný programovacímu jazyku Java, atd.

3. Tiskové sestavy v praxi

Jak již bylo naznačeno v úvodu, JasperReports je nástroj, se kterým se firma OR-CZ, ve které mi bylo umožněno vykonávat svou praxi, teprve seznamuje a zkoumá jeho možnosti. V této praktické části bych se ráda věnovala některým projektům tvorby reportů z dat OR-SYSTEMu pro samotné zákazníky, u kterých bych chtěla postupně poukázat na to, jakých možností generátoru tiskových sestav JasperReports lze využít a které byly při vývoji tvorby těchto sestav postupně objevovány.

U JasperReports byla nejprve podporována verze 4.1.2 a u iReport 4.1.1, ale protože je tvorba tiskových sestav pomocí těchto nástrojů stále ve vývoji a vzhledem k požadavkům zákazníků, bylo potřeba upgradovat na verzi 4.5.1. Hlavním důvodem k upgradování na vyšší verzi byly možnosti výstupu do Excelu, které verze 4.1.2 a 4.1.1 nepodporují. Tyto možnosti budou popsány v kapitole *3.4.1 Možnosti výstupu do Excelu*.

U většiny reportů byl jako datový zdroj použit Hibernate connection, ale toto řešení bohužel nebylo u některých sestav vyhovující. Proto byla u těchto reportů zvolena možnost využití datového zdroje JDBC connection.

3.1. Tiskové sestavy a OR-SYSTEM

„OR-SYSTEM je základním informačním systémem pro obchodní a výrobní společnosti s výrobou kusovou, malosériovou, sériovou i hromadnou. Hlavní ambicí OR-SYSTEMu je podpořit svého uživatele při udržení a dalším zvyšování jeho konkurenční výhody zabezpečením aktuálních a relevantních informací pro komplexní řízení a včasné rozhodování - vždy ve vhodném okamžiku a na správném místě.“ [12]

Obrázek 7 – Úvodní obrazovka OR-SYSTEMu
Zdroj: Printscreen - OR-SYSTEM

3.1.1. Knihovna orjasper.jar

Tiskové sestavy z dat OR-SYSTEMu jsou spouštěny pod Javou pomocí programu s názvem orjasper.jar, který umožňuje kompilaci, naplnění a export sestavy do formátů PDF, XLS, ODT, HTML a RTF. Tento program vychází ze základního programu popsaného v kapitole *2.6 Běh pod Javou* (*Ukázka kódu 28 – Java program, tvorba reportu*).

Kromě knihoven zmíněných v kapitole *2.6.1 Knihovny* je součástí programu orjasper.jar také knihovna s datovými třídami OR-SYSTEMu ors-1.0-SNAPSHOT.jar a knihovny ojdbc6.jar, orai18n.jar pro JDBC ORACLE.

V první fázi vývoje programu do něj byly přidány možnosti exportů do více formátů, kde se přípona příslušného formátu zadá jako třetí argument `argv[2]`(viz *Ukázka kódu 30 – Export do PDF* až *Ukázka kódu 34 – Export do HTML*). Pro připomenutí, prvním argumentem je název šablony s cestou, kde se šablona vyskytuje a která je OR-SYSTEMem upravena pomocí zadávací masky pro konkrétní zadání (zadávací masky

viz kapitola *3.1.2 Parametrizace tiskových sestav*), a druhým argumentem je název souboru s cestou, kam se má sestava uložit. Jak OR-SYSTEM ví, kterou sestavu má upravit? Pro každou sestavu v OR-SYSTEMu existuje přepínač, ve kterém je definován název sestavy a samozřejmě cesta k této šabloně.

Obrázek 8 – Schéma výstupu
Zdroj: Vlastní

```
if (argv[2].equals("pdf")) {
    JasperExportManager.exportReportToPdfFile(jp, argv[1]);
}
```

Ukázka kódu 30 – Export do PDF

```
if (argv[2].equals("xls")) {
    JExcelApiExporter exporterXls = new JExcelApiExporter();
    exporterXls.setParameter(JRExporterParameter.JASPER_PRINT, jp);
    exporterXls.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, argv[1]);
    exporterXls.exportReport();
}
```

Ukázka kódu 31 – Export do XLS

```
if (argv[2].equals("odt")) {
    JROdtExporter odtExporter = new JROdtExporter();
    odtExporter.setParameter(JRExporterParameter.JASPER_PRINT, jp);
    odtExporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, argv[1]);
    odtExporter.exportReport();
}
```

Ukázka kódu 32 – Export do ODT

```
if (argv[2].equals("rtf")) {
    JRRtfExporter rtfExporter = new JRRtfExporter();
    rtfExporter.setParameter(JRExporterParameter.JASPER_PRINT, jp);
    rtfExporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, argv[1]);
    rtfExporter.exportReport();
}
```

Ukázka kódu 33 – Export do RTF

```
if (argv[2].equals("htm")) {
    JRHtmlExporter htmlExporter = new JRHtmlExporter();
    htmlExporter.setParameter(JRExporterParameter.JASPER_PRINT, jp);
    htmlExporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, argv[1]);
    htmlExporter.exportReport();
}
```

Ukázka kódu 34 – Export do HTML

Protože bylo potřeba rozlišit distribuční a firemní sestavy, bylo vhodné do CLASSPATH programu zadat cesty, kde se nachází distribuční a kde firemní konfigurační soubor hibernate.properties. Proto byly v programu deklarovány dvě proměnné `orfdir` (proměnná prostředí ORFDIR pro distribuční sestavy) a `data` (proměnná prostředí DATA pro firemní sestavy), jejichž hodnota se získá pomocí metody `getenv()`. Následně jsou tyto proměnné prostředí společně s cestou, kde se konfigurační soubor nachází, přidány do CLASSPATH, k čemuž je využita třída `ClassLoader` (*Ukázka kódu 35 – Metoda přidání cesty do CLASSPATH programu*).

Soubor hibernate.properties se primárně načítá ze složky ${DATA}/jasper/cfg, pokud nebude v této složce existovat, tak se bude načítat z ${ORFDIR}/Comm/jasper/cfg. Poslední možností odkud se soubor hibernate.properties může načíst je přímo u knihovny orjasper.jar (metoda `getJarFolder()`).

```
//hibernate.properties
if (data != null && new File(data + "/jasper/cfg/hibernate.properties").exists()) {
    File file = new File(data + "/jasper/cfg/");
    URL[] urls = new URL[1];
    urls[0] = file.toURL();
    ClassLoader currentThreadClassLoader = Thread.currentThread().getContextClassLoader();
    URLClassLoader urlClassLoader = new URLClassLoader(urls, currentThreadClassLoader);
    Thread.currentThread().setContextClassLoader(urlClassLoader);

} else if (orfdir != null && new File(orfdir + "/Comm/jasper/cfg/hibernate.properties").exists()) {
    File file = new File(orfdir + "/Comm/jasper/cfg/");
    URL[] urls = new URL[1];
    urls[0] = file.toURL();
    ClassLoader currentThreadClassLoader = Thread.currentThread().getContextClassLoader();
    URLClassLoader urlClassLoader = new URLClassLoader(urls, currentThreadClassLoader);
    Thread.currentThread().setContextClassLoader(urlClassLoader);

}
```

Ukázka kódu 35 – Metoda přidání cesty do CLASSPATH programu

```
//nacteni hibernate.properties
public Parametry() throws IOException {

    String data = System.getenv("DATA");
    String orfdir = System.getenv("ORFDIR");

    if (data != null && new File(data + "/jasper/cfg/hibernate.properties").exists()) {
        prostredi = data + "/jasper/cfg/hibernate.properties";
    } else if (orfdir != null && new File(orfdir + "/Comm/jasper/cfg/hibernate.properties").exists())
        prostredi = orfdir + "/Comm/jasper/cfg/hibernate.properties";
    } else {
        prostredi = getJarFolder() + "hibernate.properties";
    }

    props.load(new FileInputStream(prostredi));
}
```

Ukázka kódu 36 – Načtení hibernate.properties

3.1.1.1. Java Database Connectivity

Pro případ využití JDBC connection byl program orjasper.jar rozšířen o tuto možnost (*Ukázka kódu 37 - JDBC*). Připojení se vytváří pomocí třídy `Connection` a metody `connectToDatabse()` na základě parametrů (klíčů) „`hibernate.connection.url`“, „hibernate`.connection.username`“ a „`hibernate.connection.password`“ ze souboru hibernate.properties. Proto bylo připojení k databázi v programu rozděleno na dvě větve, kde se jako 4. argument vyhodnocuje připojení (JDBC connection - *Ukázka kódu 38 – JDBC connection*, Hibernate connection - *Ukázka kódu 39 – Hibernate connection*).

```
static public Connection connectToDatabase() throws IOException, SQLException {

    return connectToDatabase(sJdbcURL, sUzivatel, sHeslo);

}

static private Connection connectToDatabase(String sJdbcURL, String sUzivatel, String sHeslo)
        throws SQLException, IOException {

    Parametry pJDBC = new Parametry();

    try {
        Class.forName(pJDBC.GetP("hibernate.connection.driver_class")).newInstance();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }


    sJdbcURL = pJDBC.GetP("hibernate.connection.url");
    sUzivatel = pJDBC.GetP("hibernate.connection.username");
    sHeslo = pJDBC.GetP("hibernate.connection.password");

    return DriverManager.getConnection(sJdbcURL,
            sUzivatel, sHeslo);

}
```

Ukázka kódu 37 - JDBC

```
if (argv.length > 3 && argv[3].equals("J")) {
    Connection conn = Jdbc.connectToDatabase();
    JasperPrint jp = JasperFillManager.fillReport(jr, params, conn);
```

Ukázka kódu 38 – JDBC connection

```
if (argv.length == 3 || argv[3].equals("H")) {
    Session session = NewHibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    params.put("HIBERNATE_SESSION", session);
    JasperPrint jp = JasperFillManager.fillReport(jr, params);
    session.getTransaction().commit();
```

Ukázka kódu 39 – Hibernate connection

3.1.1.2. Možnost volání Scriptletu

Pro efektivní využití Scriptletů se ukázalo, že by bylo vhodné, aby program orjasper.jar obsahoval dynamické načítání jar knihovny potřebné pro Scriptlet. Tyto knihovny se budou načítat pomocí klíče JAR_LIB obsaženého v souboru properties, který bude nést stejný název jako sestava, v níž je Scriptlet použit. Jméno tohoto properties souboru se tedy bude načítat dynamicky z obsahu šablony, ze kterého převezme její název nacházející se

za prvním výskytem slova „name“ (*Ukázka kódu 40 – Název šablony*). Název je uložen do proměnné `baseNameOfReport`, která je následně použita při načítání souboru properties vycházejícího z tohoto názvu (*Ukázka kódu 41 – Načtení souboru properties*). Poté se pomocí metody `GetP()` ke klíči JAR_LIB načte hodnota představující cestu ke knihovně pro Scriptlet, která se v případě, že je nalezena, přidá do CLASSPATH programu orjasper.jar (*Ukázka kódu 42 – Dynamické načítání JAR knihovny pro Scriptlet*).

```
//Vyhledani nazvu sablony jrxml ve vstupnim souboru jrxml
//a jeho ulozeni do promenne baseNameOfReport
FileInputStream fis = new FileInputStream(nameOfReport);
DataInputStream input = new DataInputStream(fis);
String str;
int a = 0;
int b = 0;
int c = 0;
boolean ukoncitCyklus = false;
String s3 = "name=";
String s4 = "\"";
String result = "";

while (null != ((str = input.readLine())) && !ukoncitCyklus) {
    if ((a = str.indexOf(s3, b)) > -1) {
        c = str.indexOf(s4, a + 6);
        result = str.substring(a + 7, c);
        baseNameOfReport = result;
        ukoncitCyklus = true;
    }
}
fis.close();
```

Ukázka kódu 40 – Název šablony

```
//nacteni properties souboru s názvem vytvořeným ze vstupního jrxml souboru
public Parametry2(String baseNameOfReport) throws IOException {

    if (data != null && new File(data + "/jasper/cfg/" + baseNameOfReport + ".properties").exists()) {
        prostredi = data + "/jasper/cfg/" + baseNameOfReport + ".properties";
        props.load(new FileInputStream(prostredi));
    }

    if (orfdir != null && new File(orfdir + "/Comm/jasper/cfg/" + baseNameOfReport + ".properties").exists()) {
        prostredi = orfdir + "/Comm/jasper/cfg/" + baseNameOfReport + ".properties";
        props2.load(new FileInputStream(prostredi));
    }
}

public String GetP(String P) throws IOException {

     if (props.getProperty(P) != null) {
        return props.getProperty(P);

    } else {
        return props2.getProperty(P);
    }
}
```

Ukázka kódu 41 – Načtení souboru properties

```
Parametry2 p2 = new Parametry2(baseNameOfReport);

String s1 = p2.GetP("JAR_LIB");

//nacteni .jar pro Scriptlet
if (s1 != null) {
    String s2 = p3.GetPath(s1);

    File file = new File(s2);
    URL[] urls = new URL[1];
    urls[0] = file.toURL();
    ClassLoader currentThreadClassLoader = Thread.currentThread().getContextClassLoader();
    URLClassLoader urlClassLoader = new URLClassLoader(urls, currentThreadClassLoader);
    Thread.currentThread().setContextClassLoader(urlClassLoader);
}
```

Ukázka kódu 42 – Dynamické načítání JAR knihovny pro Scriptlet

3.1.1.3. Vyhledávání subreportů a obrázků

Jelikož mohou být do sestav vkládány různé obrázky a jiné sestavy (subreporty), bylo třeba vyřešit problém s tím, kam obrázky a subreporty jednotně ukládat, aby jejich použití bylo pro uživatele co nejjednodušší.

Pro subreporty se jako nejlepší cesta zdálo být použití proměnných prostředí (ORFDIR pro distribuční úlohy a DATA pro firemní úlohy), které je znázorněno na ukázce kódu – *Proměnné prostředí*.

```
<subreportExpression>
  <![CDATA[System.getenv("DATA") == null ? "NazevSubreportu.jasper" :
  System.getenv("DATA") + "/jasper/"+" NazevSubreportu.jasper"]]>
</subreportExpression>
```

Ukázka kódu 43 – Proměnné prostředí

V příkladě je použita metoda `getenv()`, která vrací hodnotu proměnné prostředí a podmínka if-else, jejíž syntaxi lze vidět na ukázce kódu – *Podmínka if - else.*

```
(<podmínka>) ? <hodnota když true> : <hodnota když false>
```

Ukázka kódu 44 – Podmínka if - else

Pokud tedy nebude nalezena proměnná prostředí (zde DATA), je vyhledán subreport, který se nachází v adresáři společně s hlavním reportem, jinak se subreport bere ze zadané cesty ($DATA/jasper/NazevSubreportu.jasper).

Ale ukázalo se, že toto řešení není pro uživatele zcela ideální, že existuje i jednodušší způsob, který pro ně bude vhodnější. Proto byly do CLASSPATH programu orjasper.jar přidány cesty, kam se reporty a jejich subreporty budou jednotně ukládat (opět jsou zde rozlišeny firemní a distribuční sestavy). Způsob přidání těchto cest do CLASSPATH programu je obdobný jako v ukázce kódu – *Metoda přidání cesty do CLASSPATH programu.*

Do šablony tedy není potřeba zadávat složitější podmínku s proměnnou prostředí, ale stačí zadat jen název subreportu, který se nachází ve složce System.getenv("DATA") /jasper nebo System.getenv("ORFDIR") /jasper (*Ukázka kódu 45 – Cesta k subreportu*).

Podobným způsobem je to řešeno i s obrázky. Obrázky se jednotně ukládají do složky images, která musí být vytvořena ve složce System.getenv("ORFDIR") /jasper a System.getenv("DATA") /jasper. Proto u obrázků nestačí zadat pouze název obrázku, ale i tuto složku images (*Ukázka kódu 46 – Cesta k obrázku*).

```
<subreportExpression>
  <![CDATA["NazevSubreportu.jasper"]]>
</subreportExpression>
```

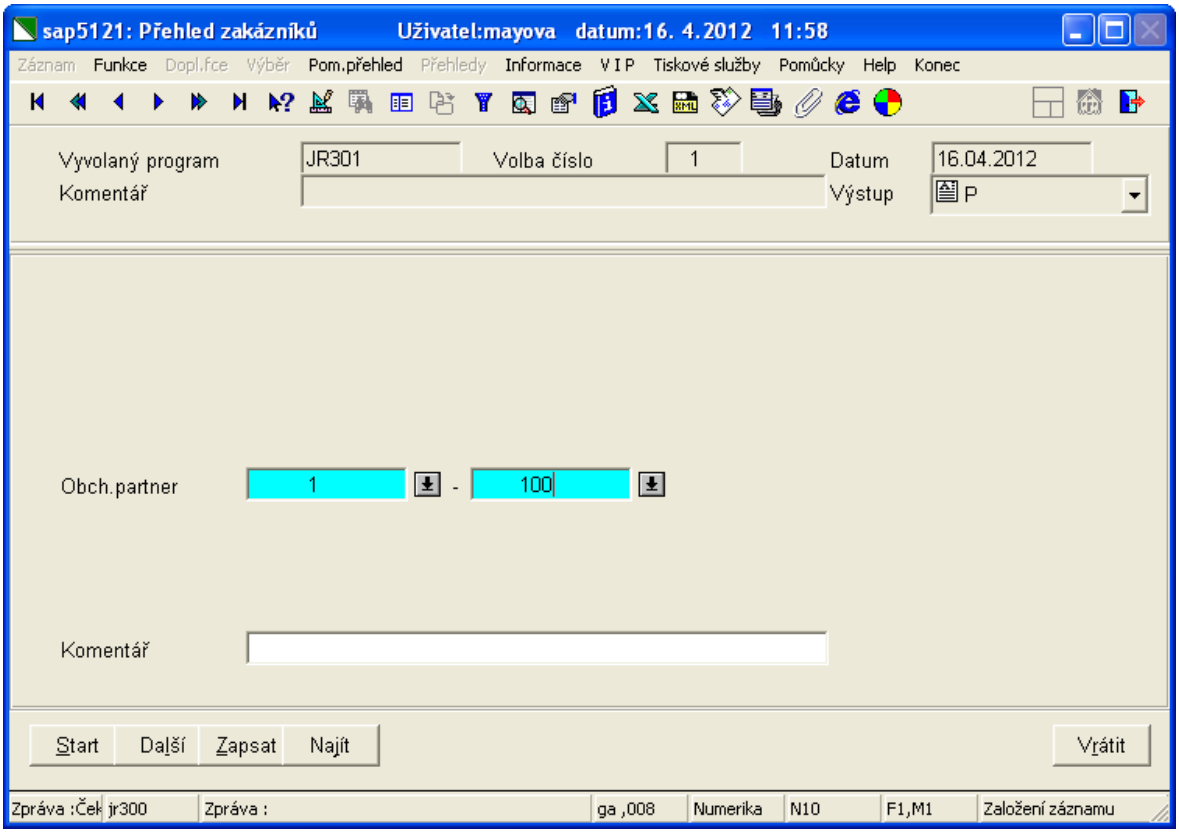
Ukázka kódu 45 – Cesta k subreportu

```
<imageExpression>
  <![CDATA["images/NazevObrazku.png"]]>
</imageExpression>
```

Ukázka kódu 46 – Cesta k obrázku

3.1.2. Parametrizace tiskových sestav

Jednou z nejdůležitějších vlastností tvorby reportů z dat OR-SYSTEMu je možnost jejich parametrizace. Jak již bylo řečeno v kapitole *2.3.5 Parametry,* je možné si v šabloně definovat své vlastní parametry. Tyto parametry jsou předávány z dat OR-SYSTEMu na základě zadávací masky (*Obrázek 9 – Zadávací maska*), kde je defaultní hodnota parametru v šabloně nahrazena hodnotou zadanou v této zadávací masce. Zadávací maska na obrázku patří k ukázce reportu popsanému v kapitole *2.4 Tvorba reportu.* Díky parametrizaci se dotaz stává pružnějším a není tak potřeba měnit šablonu při každé změně kritéria, které je definováno parametrem.

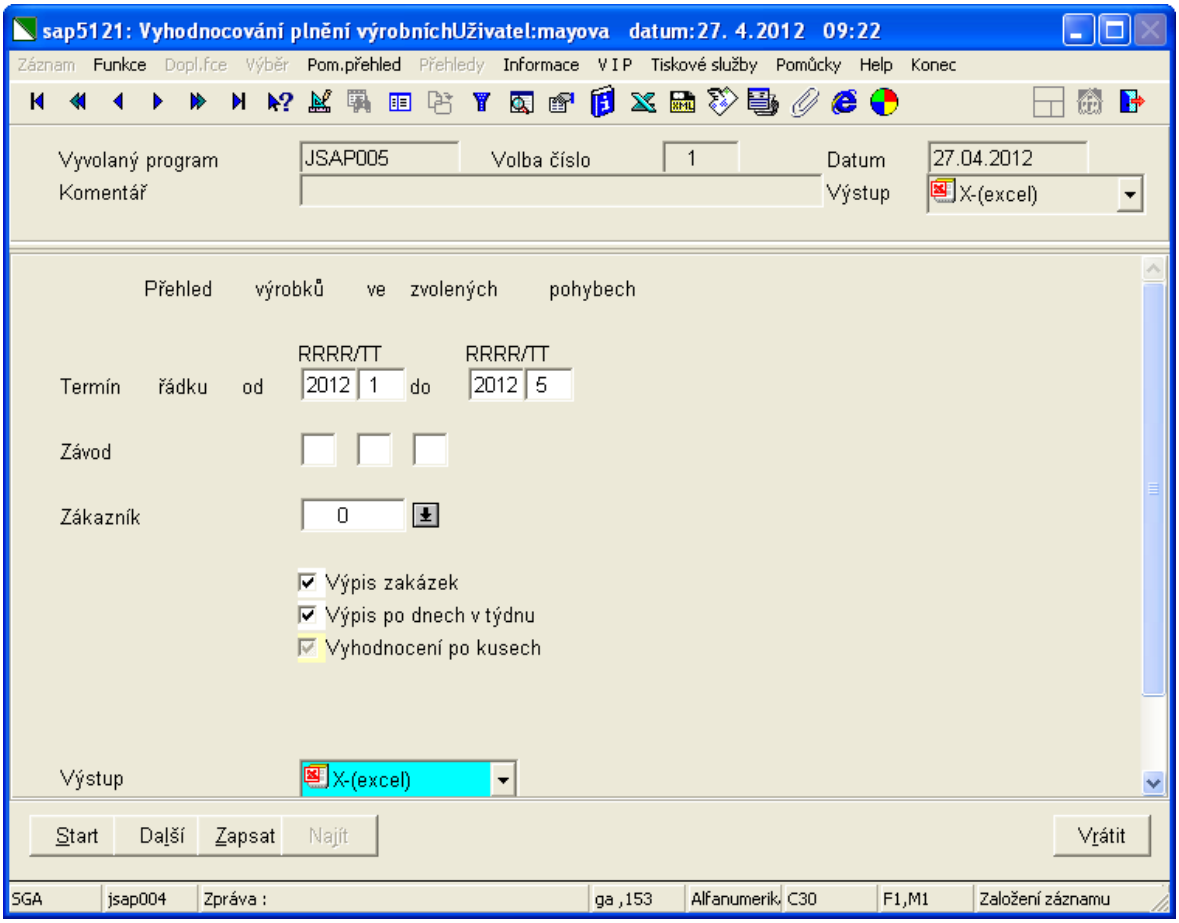


Obrázek 9 – Zadávací maska
Zdroj: Printscreen - OR-SYSTEM

3.1.2.1. Dynamická tvorba dotazu

JasperReports umožňuje díky parametrům dynamicky vytvářet dotazy. Tato výhoda se ukázala jako velice výhodná, je-li po sestavě žádáno, aby uživateli dovolila omezit určitá kritéria zadáním parametrů, ale aby se počet těchto parametrů přizpůsoboval potřebám uživatele. Což znamená, že uživateli je pouze nabídnuta možnost parametrizace sestavy dle požadovaných kritérií a jen na něm záleží, jestli bude tisk sestavy těmito parametry nějak omezovat, nebo vytiskne sestavu v plném rozsahu.

Například na obrázku – *Zadávací maska – dynamická tvorba dotazu* je možné pomocí parametrů „Závod“ a „Zákazník“ sestavu parametrizovat. U parametru „Závod“ uživatel může zadat žádný, nebo jeden, dva či tři parametry. Podle toho, kolik jich uživatel zadá, se následně v sestavě dynamicky vytvoří WHERE podmínka s danými parametry. U parametru „Zákazník“ je to naprosto totožné, uživatel může zvolit tisk sestavy jen pro konkrétního zákazníka.



Obrázek 10 – Zadávací maska – dynamická tvorba dotazu
Zdroj: Printscreen - iReport

3.2. Pilotní projekt JasperReports

Tento report byl tvořen za účelem, aby si zákazníci mohli prohlédnout, v jakém stavu se nachází jejich zakázka, popřípadě, aby jim obchodníci na jejich žádost mohli informace o zakázce sdělit či zaslat e-mailem v požadovaném formátu PDF.

V této kapitole bych chtěla poukázat na to, k čemu slouží subreporty a jaké je jejich účelové použití v praxi.

Prvním, tak zvaným pilotním projektem byla sestava zobrazující po zadání čísla zakázky uživatelem aktuální přehled o stavu této zakázky. Sestava má ukazovat jeden z těchto stavů:

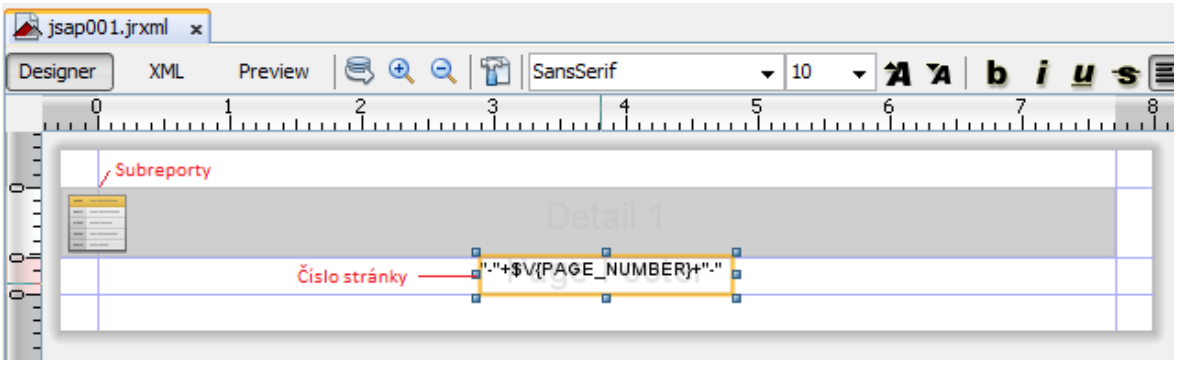
1. V systému neexistuje kupní smlouva ani nabídka

2. V systému existuje nabídka
3. V systému existuje kupní smlouva a není uvolněna
4. V systému existuje uvolněná kupní smlouva

Podstata této tiskové sestavy spočívá v použití subreportů, které se tisknou na základě určité podmínky. Celý report se tedy skládá z jednoho hlavního reportu (jsap001.jrxml) a devíti subreportů. Struktura sestavy vypadá takto:

- **jsap001.jrxml** (test existence uvolněné kupní smlouvy)
  - **jsap001_01.jrxml** (subreport - uvolněná kupní smlouva)
    - **jsap001_01_01.jrxml** (subreport - řádky uvolněné kupní smlouvy)
  - **jsap001_02.jrxml** (subreport - test existence neuvolněné kupní smlouvy)
    - **jsap001_02_01.jrxml** (subreport - neuvolněná kupní smlouva)
      - **jsap001_02_01_01.jrxml** (subreport - řádky neuvolněné kupní smlouvy)
    - **jsap001_02_02.jrxml** (subreport - test existence nabídky)
      - **jsap001_02_02_01.jrxml** (subreport - nabídka)
        - **jsap001_02_02_01_01.jrxml** (subreport - řádky nabídky)
      - **jsap001_02_02_02.jrxml** (subreport - neexistuje kupní smlouva ani nabídka)

V hlavním reportu jsap001.jrxml v sekci Detail jsou vloženy pouze dvě další sestavy (subreporty) – jsap001_01.jrxml a jsap001_02.jrxml a počet stránek v sekci Page Footer pomocí systémové proměnné `$V{PAGE_NUMBER}`.



Obrázek 11 – Šablona jsap001.jrxml
Zdroj: Printscreen - iReport

Na obrázku – *Šablona jsap001.jrxml* je vidět jen jeden subreport a dvě sekce - Detail a Page Footer (ostatní sekce jsou odstraněné, protože nebyly potřeba). Jeden subreport je vidět z toho důvodu, že se subreporty vzájemně překrývají (jsou dané přes sebe).

Ukázka kódu 47 – HQL dotaz znázorňuje vytvořený dotaz prostřednictvím jazyka HQL. Tabulka AKA (zde třída) je tabulka záhlaví zakázky, `id.akaAnr` a `id.akaAa` jsou primárními klíči (pokud tabulka obsahuje složený primární klíč, přistupuje se k němu pomocí syntaxe id, pak následuje malými písmeny název tabulky a potom název sloupce začínající velkým písmenem). Atribut `id.akaAnr` představuje číslo zakázky, které do sestavy zadá uživatel přes vstupní parametr (`$P{01}`), `id.akaAa` je typ zakázky (zde kupní smlouva (`akaAa=0`)) a `akaKzuv` je atribut pro uvolněnou kupní smlouvu (nabývá buď hodnoty „J“, nebo „N“).

```
<queryString language="hql">
      <![CDATA[select count(*) as pocet
      from Aka
      where id.akaAnr=$P{01} and id.akaAa=0 and akaKzuv='J']]>
</queryString>
```

Ukázka kódu 47 – HQL dotaz

Tento hlavní report testuje, zda bude nalezena uvolněná kupní smlouva se zadaným číslem `akaAnr`. Na základě vyhodnocení tohoto dotazu se bude tisknout buď jeden subreport, nebo druhý. Toto se řeší pomocí výrazu „`printWhenExpression`“. U každého ze subreportů, jak již bylo řečeno v kapitole *2.3.5 Parametry* je také třeba předat parametry mezi daným subreportem a hlavním reportem. U obou subreportů to bude systémový parametr HIBERNATE_SESSION, jelikož byl použit jako datový zdroj Hibernate a parametr čísla zakázky `$P{Anr}`, který je použit jako parametr v hlavním reportu i v subreportech (*Ukázka kódu 48 – Použití subreportů*).

Pokud tedy bude počet uvolněných kupních smluv větší než nula, vytiskne se záhlaví uvolněné kupní smlouvy (*Ukázka kódu 49 – Záhlaví kupní smlouvy*) a k němu příslušné řádky (*Ukázka kódu 50 – Řádky kupní smlouvy*), které jsou do sestavy vloženy jako subreport. Jak tyto sestavy vypadají v grafickém návrháři iReport je znázorněno na obrázku – *Hlavička uvolněné kupní smlouvy* a na obrázku – *Řádky uvolněné kupní smlouvy*. Mezi záhlavím a řádky uvolněné kupní smlouvy je také třeba předat parametry.

Stejně jako u předchozího případu to bude systémový parametr HIBERNATE_SESSION a číslo zakázky `Anr`.

Jestliže se žádná uvolněná kupní smlouva nenajde, bude následovat další rozhodovací subreport obsahující opět dva subreporty. Princip postupu je stále stejný, jen se tentokrát bude testovat, jestli v databázi existuje neuvolněná kupní smlouva (dotaz bude stejný, jen `akaKzuv` bude různé od 'J'). V případě, že nějaká neuvolněná kupní smlouva existuje, vytiskne se záhlaví neuvolněné kupní smlouvy a k němu příslušné řádky, které jsou opět vloženy jako subreport. Pokud žádná neuvolněná kupní smlouva nalezena nebude, dojde na poslední rozhodovací subreport, který otestuje existenci nabídky. Při existenci nabídky se opět vytiskne záhlaví nabídky a k ní příslušné řádky, které jsou vložené jako subreport, nebo se naopak vytiskne zpráva, že v systému neexistuje kupní smlouva ani nabídka se zadaným číslem.

```
<subreport>
      <reportElement x="-20" y="0" width="575" height="38">
            <printWhenExpression>
                  <![CDATA[$F{pocet}.intValue() > 0]]>
            </printWhenExpression>
      </reportElement>
      <subreportParameter name="HIBERNATE_SESSION">
            <subreportParameterExpression>
                  <![CDATA[$P{HIBERNATE_SESSION}]]>
            </subreportParameterExpression>
      </subreportParameter>
      <subreportParameter name="Anr">
            <subreportParameterExpression>
                  <![CDATA[$P{01}]]>
            </subreportParameterExpression>
      </subreportParameter>
      <subreportExpression>
            <![CDATA["jsap001_01.jasper"]]>
      </subreportExpression>
</subreport>
<subreport>
      <reportElement x="-20" y="0" width="575" height="38">
            <printWhenExpression>
                  <![CDATA[$F{pocet}.intValue() == 0]]>
            </printWhenExpression>
      </reportElement>
      <subreportParameter name="HIBERNATE_SESSION">
            <subreportParameterExpression>
                  <![CDATA[$P{HIBERNATE_SESSION}]]>
            </subreportParameterExpression>
      </subreportParameter>
      <subreportParameter name="Anr">
            <subreportParameterExpression>
                  <![CDATA[$P{Anr}]]>
            </subreportParameterExpression>
```

```
      </subreportParameter>
      <subreportExpression>
            <![CDATA["jsap001_02.jasper"]]>
      </subreportExpression>
</subreport>
```

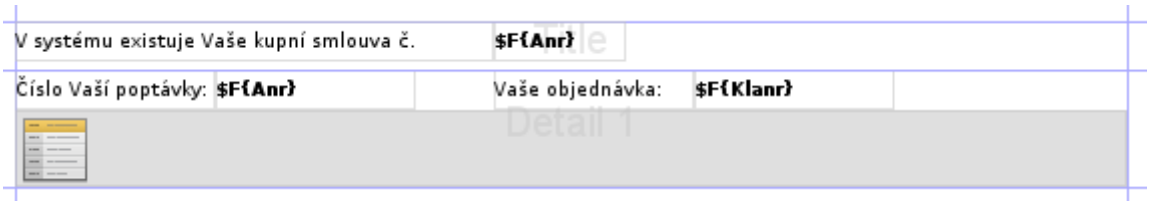
Ukázka kódu 48 – Použití subreportů

```
<queryString language="hql">
      <![CDATA[from Aka
      where id.akaAa = 0 and id.akaAnr = $P{akaAnr}]]>
</queryString>
```

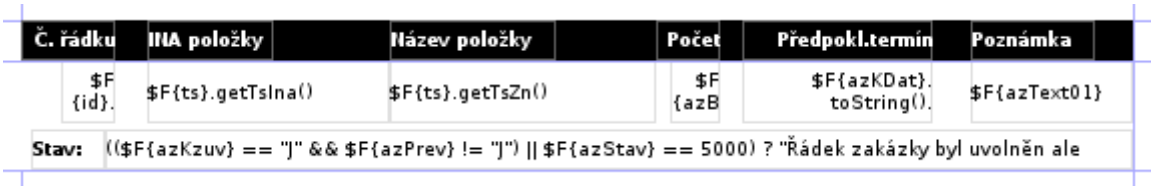
Ukázka kódu 49 – Záhlaví kupní smlouvy

```
<queryString language="hql">
      <![CDATA[from Az
      where id.azAa = 0 and id.azAnr = $P{azAnr}]]>
</queryString>
```

Ukázka kódu 50 – Řádky kupní smlouvy



Obrázek 12 – Hlavička uvolněné kupní smlouvy
Zdroj: Printscreen - iReport



Obrázek 13 – Řádky uvolněné kupní smlouvy
Zdroj: Printscreen - iReport

Kromě subreportů je zde využito prvků, jako jsou styly a rámeček (*Ukázka kódu 51 – Styl a rámeček*

```
...
<style name="radky">
  <conditionalStyle>
    <conditionExpression>
      <![CDATA[$V{REPORT_COUNT}.intValue()%2 == 0]]>
    </conditionExpression>
    <style mode="Opaque" backcolor="#999999"/>
  </conditionalStyle>
</style>
...
<columnHeader>
  <band height="20" splitType="Stretch">
```

```
    <frame>
      <reportElement mode="Opaque" x="0" y="0" width="555"
      height="20" forecolor="#FFFFFF" backcolor="#000000"/>
      ...
    </frame>
  </band>
</columnFooter>
<detail>
  <band height="54" splitType="Stretch">
    <frame>
      <reportElement style="radky" mode="Opaque" x="0" y="0"
      width="555" height="54"/>
      ...
    </frame>
  </band>
</detail>
</jasperreports>
```

Ukázka kódu 51 – Styl a rámeček

Ukázka výsledné sestavy exportované do PDF je obsažena v příloze B, *str. 88*.

3.3. Využití Scriptletů

Jak již sám název napovídá, v této kapitole budou představeny Scriptlety a jejich praktické využití v tiskových sestavách.

Základem tohoto reportu je přenos položek materiálu z dat OR-SYSTEMu (Oracle) do databáze PC Schematic[1] (MS Access) vyvolaný prostřednictvím Scriptletu (Java program pro přenos položek materiálu) a export informací o tomto přenosu do formátu PDF (viz dále). Nastavení vazby a přístupu k oběma databázím bude definováno v souboru properties (z důvodu možnosti doplnění nové vazby a aktualizace stávající). Tento properties soubor bude nést název tiskové sestavy pojmenované jtm001. Jednotlivé sloupce se budou přenášet takto:

OR-SYSTEM		**PC SCHEMATIC**
ts_ina	→	SKLADčíslo
ts matutek	→	OBJčíslo
ts_zn	→	TYP
ts_bez1	→	POPIS
ts_bez2	→	POZNÁMKA

[1] více na http://www.pcschematic.com/en/index.htm

ts_matosnova → KATALOG
ts_text → VÝROBCE

Sestava bude obsahovat dva parametry, a to volbu zapisování nových položek a aktualizaci stávajících. Tyto parametry se následně předají do metody `startZpracuj()`, která bude obsažena ve třídě `PCS_Scriptlet`. Další metoda v této třídě `getCounts()` bude vracet pole čísel Integer (přečtené, aktualizované, nové záznamy).

```
$P{REPORT_SCRIPTLET}.getCounts()[0]
$P{REPORT_SCRIPTLET}.getCounts()[1]
$P{REPORT_SCRIPTLET}.getCounts()[2]
```

V souboru properties s názvem jtm001.properties bude definováno připojení jak k databázi OR-SYSTEMu, tak k databázi PC Schematic. Navíc zde budou obsaženy jednotlivé SQL dotazy pro přečtení, aktualizaci a přidání nových záznamů (*Ukázka kódu 52 – SQL dotazy*). Načtení tohoto souboru se provádí na stejném principu, jako tomu bylo u souboru hibernate.properties (viz *Ukázka kódu 36 – Načtení hibernate.properties*).

Připojení k oběma databázím je obdobné jako v ukázce kódu - *JDBC* (Oracle - *Ukázka kódu 53 – Připojení k databázi Oracle* i MS Access - *Ukázka kódu 54 – Připojení k databázi Access*).

```
#konfigurace připojení k OR-SYSTEMu
URL_JDBC=jdbc\:oracle\:thin\:@IP adresa\:port\:jméno služby
USER_NAME=username
#konfigurace připojení k PC Schematic
URL_ODBC=jdbc\:odbc\:PCS
SQL01=SELECT      substr(ts_ina,1,3)||'-'||
                  substr(ts_ina,4,2)||'-'||substr(ts_ina,6,4),\
                  ts_matutek, \
                  ts_zn     , \
                  ts_bez1   , \
                  ts_bez2   , \
                  ts_matosnova,\
                  ts_text      \
                  FROM ts      \
      WHERE ts_ch\=9 \
      AND
      TRANSLATE(ts_ina,'0123456789','9999999999')\='999999999'
SQL02=INSERT INTO pcs \
      (SKLAD\u010D\u00EDslo,\
      OBJ\u010D\u00EDslo  ,\
      TYP       ,\
      POPIS     ,\
      POZN\u00C1MKA  ,\
      KATALOG   ,\
      V\u00DDROBCE   )\
      values (?,?,?,?,?,?,?)
```

```
SQL03=UPDATE PCS SET \
      OBJ\u010D\u00EDslo \= ?     , \
      TYP       \= ?     , \
      POPIS     \= ?     , \
      POZN\u00C1MKA \= ?     , \
      KATALOG   \= ?     , \
      V\u00DDROBCE   \= ?       \
      WHERE SKLAD\u010D\u00EDslo =  ?
SQL04=SELECT COUNT(*) \
      FROM PCS \
      WHERE SKLAD\u010D\u00EDslo = ?
```

Ukázka kódu 52 – SQL dotazy

```
static public Connection connectToDatabase() throws SQLException, IOException, URISyntaxException {

    return connectToDatabase(sJdbcURL, sUzivatel, sHeslo);

}

static private Connection connectToDatabase(String sJdbcURL,
        String sUzivatel,
        String sHeslo)
        throws SQLException, IOException, URISyntaxException {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver").
                newInstance();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }

    Parametry pJDBC = new Parametry();
    sJdbcURL = pJDBC.GetP("URL_JDBC");
    sUzivatel = pJDBC.GetP("USER_NAME");
    sHeslo = pJDBC.GetP("USER_NAME");

    return DriverManager.getConnection(sJdbcURL,
            sUzivatel, sHeslo);
}
```

Ukázka kódu 53 – Připojení k databázi Oracle

```
static public Connection connectToDatabase() throws SQLException, IOException, URISyntaxException {

    return connectToDatabase(PCS);
}

static public Connection connectToDatabase(String PCS) throws SQLException, IOException, URISyntaxException {


    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").
                newInstance();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }

    Parametry pODBC = new Parametry();
    PCS = pODBC.GetP("URL_ODBC");

    return DriverManager.getConnection(PCS);
}
```

Ukázka kódu 54 – Připojení k databázi Access

Následně se vytvoří třída `OrPcs`, v níž bude zahrnuta metoda `zpracuj(String nove, String aktualizovane)`. Parametry obsažené v této metodě mohou nabývat buď hodnoty „J“ nebo „N“. Metoda `zpracuj()` načte SQL dotazy a na základě zadaných parametrů, podle kterých se provede, nebo naopak neprovede zápis nových a aktualizace stávajících záznamů, vypočítá přečtené, aktualizované a nové záznamy.

Nakonec může být vytvořena třída `PCS_Scriptlet` s již zmiňovanými metodami `startZpracuj()` a `getCounts()` (*Ukázka kódu 55 – Třída PCS_Scriptlet*).

```
public class PCS_Scriptlet extends JRDefaultScriptlet {

    public OrPcs op = new OrPcs();

    public Integer[] getCounts() throws JRScriptletException {

        Integer[] counts = new Integer[3];
        counts[0] = op.zPrectene;
        counts[1] = op.zNove;
        counts[2] = op.zAktualizovane;
        return counts;
    }

    public String startZpracuj(String nove, String aktualizovane) throws
            JRScriptletException, IOException, SQLException, URISyntaxException {

        op.zpracuj(nove, aktualizovane);

        return "";
    }
}
```

Ukázka kódu 55 – Třída PCS_Scriptlet

Aby byl Scriptlet spustitelný z grafického návrháře iReport, přidá se celý projekt (TM-ORSystem-PCS) do CLASSPATH iReportu. Jestliže má být spustitelný z OR-SYSTEMu je třeba tento projekt zařadit mezi podpůrné knihovny programu orjasper.jar. V šabloně v elementu `<jasperReport>` se definuje třída `PCS_Scriptlet` jako `scriptletClass="cz.tm.integrace.pcs.PCS_Scriptlet"`. Poté mohou být v sestavě pomocí systémového parametru `$P{REPORT_SCRIPTLET}` použity potřebné metody z této třídy. Metoda `startZpracuj()` bude v reportu obsahovat dva parametry `$P{01}` a `$P{02}` nabývající hodnot „J“, nebo „N“. Pro zjištění jména toho, kdo sestavu spustil, se použije metoda `getenv()` a proměnná prostředí `USID`, pokud ta nebude nalezena, použije se proměnná `USERNAME`.

```
...
<queryString language="hql">
  <![CDATA[from Ts where tsCh=9]]>[2]
</queryString>
...
<pageHeader>
  <band height="20" splitType="Stretch">
    <textField>
      <reportElement x="0" y="0" width="610" height="20"/>
      <textElement verticalAlignment="Middle"/>
```

[2] Ts - Položky materiálu, tsCh = 9 - výrobní položky

```
      <textFieldExpression>
        <![CDATA[$P{REPORT_SCRIPTLET}.startZpracuj($P{01},$P{02})]]>
      </textFieldExpression>
    </textField>
  </band>
</pageHeader>
<columnHeader>
...
    <textField>
      <reportElement x="164" y="0" width="446" height="20"/>
      <textElement verticalAlignment="Middle">
        <font pdfEncoding="Cp1250"/>
      </textElement>
      <textFieldExpression>
        <![CDATA[System.getenv("USID")==null?System.getenv("USERNAME"):
        System.getenv("USID")]]>
      </textFieldExpression>
    </textField>
    ...
    <textField>
      <reportElement x="164" y="40" width="446" height="20"/>
      <textElement verticalAlignment="Middle"/>
      <textFieldExpression>
        <![CDATA[$P{REPORT_SCRIPTLET}.getCounts()[0]]]>
      </textFieldExpression>
    </textField>
    <textField>
      <reportElement x="164" y="60" width="446" height="20"/>
      <textElement verticalAlignment="Middle"/>
      <textFieldExpression>
        <![CDATA[$P{REPORT_SCRIPTLET}.getCounts()[1]]]>
      </textFieldExpression>
    </textField>
    <textField>
      <reportElement x="164" y="80" width="446" height="20"/>
      <textElement verticalAlignment="Middle"/>
      <textFieldExpression>
        <![CDATA[$P{REPORT_SCRIPTLET}.getCounts()[2]]]>
      </textFieldExpression>
    </textField>
    ...
```

Ukázka kódu 56 – Scriptlet

Ukázka výsledné sestavy exportované do formátu PDF je obsažena v příloze C, *str. 89*.

3.4. Možnosti seskupování dat v sestavě

Tento report je ukázkou seskupování dat dle skladů a čísel dodavatelů. Kromě toho, je zde také využita možnost dynamického načítání dotazu pomocí parametrů. Při tvorbě této sestavy se zjistily některé nedostatky verzí 4.1.2 a 4.1.1 oproti nejnovější verzi 4.5.1. Tyto nedostatky se týkaly konkrétně možností při výstupu do Excelu, jako je například

seskupování (rozbalování a sbalování určitých částí pomocí tlačítek „+“ a „-“). Také je to ukázkový report, ve kterém nebyl jako datový zdroj využit Hibernate connection, ale JDBC connection.

V této sestavě se vyskytují informace týkající se nakoupeného materiálu a jeho cen u jednotlivých položek materiálu. Ale také je v ní poskytnuto součtování za jednotlivé dodavatele materiálu a sklady, na které byl materiál přijímán.

Sestava bude obsahovat:

- omezení období
- možnost volby všech nebo omezení 1 - 6 dodavatelů
- možnost volby všech nebo omezení 1 - 6 skladů, na kterých v daném období byly přijímány položky
- za dodavatele a sklad soupis položek - podle dodavatele, který je přiřazen u položky
- u položek - příjem za zadané období + dodavatelská cena + poslední nákupní cena
- možnost tisku pouze součtování hodnoty nakoupeného materiálu

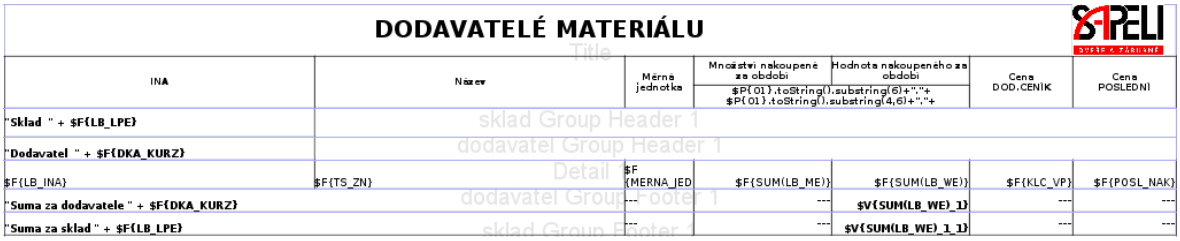
Tabulka 1 – Vzhled sestavy

sklad 100100					
dodavatel HOBES		množství nakoupené za období od - do	hodnota nakoupeného za období od - do	cena DOD.CENÍK	cena POSLEDNÍ
INA	zámek obyč				
INA	zámek WC				
INA	zámek vložk.				
INA	zámek protipožár				
	suma:			--	–

Zdroj: Vlastní

V tabulce – *Vzhled sestavy* je vidět, že položky materiálu se budou seskupovat podle skladů a dodavatelů. V reportu bude v něm obsaženo celkem 17 parametrů: 2 na zadání období od – do (`$P{01}`, `$P{02}`), 12 pro sklady a dodavatele (`$P{03}`-`$P{14}`), 1 pro tisk pouze součtování (potlačí se sekce Detail (`$P{15}`)) a 2 pro dynamické doplnění WHERE podmínky do dotazu (`$P{S01}`, `$P{S02}`). Nyní se může vytvořit SQL dotaz, který bude obsahovat všechny atributy, pole, která se budou do sestavy vkládat: `LB_INA` (identifikační číslo položky), `TS_ZN` (název položky), `MERNA_JEDNOTKA`, `LB_ME` (množství), `LB_WE` (hodnota), `KLC_VP` (dodavatelská cena), `POSL_NAK` (poslední nákupní cena), `LB_LPE` (sklad), `DKA_KURZ` (zkratka

dodavatele). Sestava také bude obsahovat 2 uživatelské proměnné, a to pro součtování hodnoty nakoupeného materiálu za jednotlivé dodavatele (`$V{SUM(LB_WE)_1}`) a sklady (`$V{SUM(LB_WE)_1_1}`)



Obrázek 14 – Šablona sestavy
Zdroj: Printscreen - iReport

Potlačení tisku sekce Detail (výpis položek materiálu) přes parametr `$P{15}` se vyřeší pomocí výrazu, kdy se tato sekce bude tisknout v případě, že parametr `$P{15}` bude roven hodnotě „J“, což je znázorněno v ukázce kódu – *Potlačení tisku.*

```
…
<detail>
  <band height="25" splitType="Stretch">
    <printWhenExpression><![CDATA[$P{15}=="J"]]></printWhenExpression>
    …
```

Ukázka kódu 57 – Potlačení tisku

Protože ne vždy se sekce Detail bude tisknout, bylo by zbytečné, kdyby se tiskly všechny nadpisy (konkrétně pole INA a NÁZEV). Proto je tisk těchto polí potlačen, pokud se netiskne sekce Detail (`$P{15}=="J"`), a to pomocí výrazu `<printWhenExpression>`.

Z obrázku – *Šablona sestavy* lze vyčíst, že všechna pole jsou ohraničena černým rámečkem. Toto je řešeno pomocí použití stylu „`rámeček`“ (*Ukázka kódu 58 – Styl „rámeček“*).

```
<style name="ramecek">
  <box topPadding="0" leftPadding="0" bottomPadding="0" rightPadding="0">
    <pen lineWidth="0.5"/>
    <topPen lineWidth="0.5"/>
    <leftPen lineWidth="0.5"/>
    <bottomPen lineWidth="0.5"/>
    <rightPen lineWidth="0.5"/>
  </box>
</style>
```

Ukázka kódu 58 – Styl „rámeček“

Jak již bylo řečeno, v této sestavě je využito také dynamické tvorby dotazu pro možnost doplnění WHERE podmínky v případě, pokud uživatel bude chtít omezit sklady (*Ukázka kódu 59 – Omezení skladů*) nebo dodavatele (*Ukázka kódu 60 – Omezení dodavatelů*).

Pokud nebude do parametru `$P{03}` zadáno nic, do dotazu se také nic nedoplní, ale jestliže `$P{03}` bude obsahovat hodnotu, do dotazu se přidá podmínka „`AND lb_lpe IN('"+$P{03}.trim()+"'")`“. Poté se postupně vyhodnocují následující parametry – jestliže parametr není mezera, potom se doplní spolu s předchozím parametrem do podmínky.

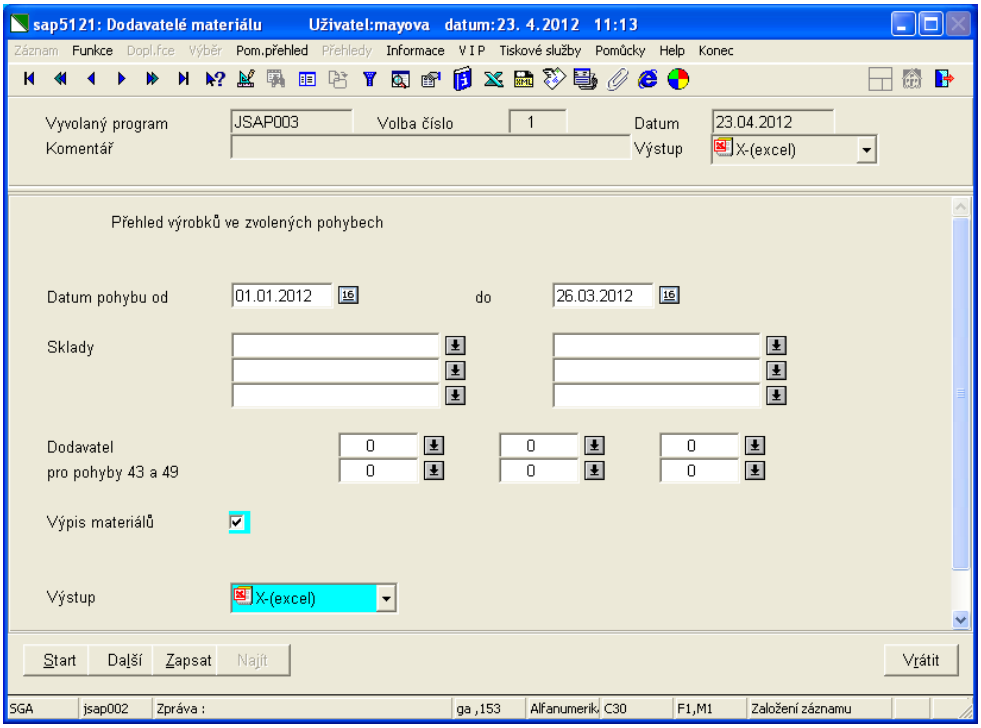
```
<parameter name="S01" class="java.lang.String" isForPrompting="false">
  <parameterDescription>
    <![CDATA[VycetSkladu]]>
  </parameterDescription>
  <defaultValueExpression>
    <![CDATA[$P{03}.trim() == "" ? "" :
            "AND lb_lpe IN ('"+ $P{03}.trim()+"'"+
            ($P{04}.trim() != "" ? ",'" + $P{04}.trim()+"'":"")+
            ($P{05}.trim() != "" ? ",'" + $P{05}.trim()+"'":"")+
            ($P{06}.trim() != "" ? ",'" + $P{06}.trim()+"'":"")+
            ($P{07}.trim() != "" ? ",'" + $P{07}.trim()+"'":"")+
            ($P{08}.trim() != "" ? ",'" + $P{08}.trim()+"'":"")+")"]]>
  </defaultValueExpression>
</parameter>
```

Ukázka kódu 59 – Omezení skladů

```
<parameter name="S02" class="java.lang.Integer" isForPrompting="false">
  <parameterDescription>
    <![CDATA[VycetDodavatelu]]>
  </parameterDescription>
  <defaultValueExpression>
    <![CDATA[$P{09}.trim() == "" ? "" :
            "AND dka_nr IN ("+ $P{09}.trim()+
            ($P{10}.trim() != "" ? "," + $P{10}.trim():"")+
            ($P{11}.trim() != "" ? "," + $P{11}.trim():"")+
            ($P{12}.trim() != "" ? "," + $P{12}.trim():"")+
            ($P{13}.trim() != "" ? "," + $P{13}.trim():"")+
            ($P{14}.trim() != "" ? "," + $P{14}.trim():"")+")"]]>
  </defaultValueExpression>
</parameter>
```

Ukázka kódu 60 – Omezení dodavatelů

Na obrázku – *Maska k sestavě jsap003* lze vidět masku k této sestavě s možnými parametry a také s možností výběru výstupu.



Obrázek 15 – Maska k sestavě jsap003
Zdroj: Printscreen - OR-SYSTEM

3.4.1. Možnosti výstupu do Excelu

V této sestavě se využilo hned několik vlastností, které nabízí JasperReports pro přehlednější výstup do Excelu. Například vlastnost „`net.sf.jasperreports.export.xls.white.page.background`“, která může nabývat hodnoty `false`, nebo `true`, zajišťuje, jestli pozadí v Excelu bude bílé (`true`), nebo mřížkované (`false`).

Další vlastnost zjišťující typ buněk dle datového typu polí je „`net.sf.jasperreports.export.xls.detect.cell.type`“ (také nabývá hodnot `true` a `false`).

V této sestavě byla využita i vlastnost „`net.sf.jasperreports.export.xls.freeze.row.edge`“, která „zmrazí“ v Excelu určité řádky, což znamená, že se budou posouvat zároveň s posuvníkem. Na rozdíl od předchozích vlastností, které se definují hned na začátku šablony v elementu `<property>`, se tato definuje v dané sekci u některého z polí, které se v této sekci

objevují. Ale neplatí jen pro toto pole, platí pro celý řádek, kde se pole vyskytuje. Tato vlastnost může být nastavena na hodnotu `Bottom`, nebo `Top`. Hodnota `Bottom` říká, že celý řádek je posledním řádkem, který je „zmrazený“ (tzn., pokud by nad tímto řádkem byly ještě nějaké řádky, tak by byly také „zmrazené“ a naopak všechny řádky pod tímto řádkem už „zmrazené“ nejsou). Hodnota `Top` znamená, že řádek, u kterého je tato hodnota nastavena, je prvním „nezmrazeným“ řádkem.

A konečně vlastnost, která dokáže v Excelu seskupovat data, „`net.sf.jasperreports.export.xls.row.outline.level.{úroveň seskupení}`“. Tato vlastnost může nabývat hodnoty `Body` (řádky, které mají být v seskupení zahrnuty), nebo `End` (řádek, kde se má seskupení ukončit).

3.4.1.1. Seskupování v Excelu

Dříve, než se nadefinuje vlastnost seskupování pro Excel, je potřeba v šabloně vytvořit skupiny a k nim příslušné sekce. V této sestavě se budou seskupovat data dle skladů a dodavatelů, kde o skladech by se dalo hovořit jako o primárních skupinách a o dodavatelích jako o podskupinách. U dodavatelů se budou seskupovat položky materiálu a u skladů dodavatelé. Jelikož má být v sestavě součtování hodnoty nakoupeného materiálu za dodavatele a sklady, tak se po vytvoření skupin definují jejich proměnné pro součtování. U první úrovně možnosti seskupení v Excelu (sklad) tedy bude vidět jen číslo skladu v sekci Group Header a sumace za sklad v sekci Group Footer, u druhé úrovně (dodavatel) se po seskupení zobrazí číslo skladu a k němu příslušní dodavatelé v sekci Group Header a jejich sumace v sekci Group Footer. Pro dodavatele tak hodnota `Body` musí být v části Detail a hodnota `End` v sekci Group Footer, kde je obsažena sumace za dodavatele. Pro sklady pak bude hodnota `Body` obsažena v sekci Group Header, kde se vyskytuje název dodavatele a hodnota `End` v sekci Group Footer obsahující součtování za daný sklad.

Problém by mohl nastat, pokud by se část Detailu netiskla. V tomto případě by seskupení dle dodavatelů v Excelu bylo naprosto zbytečné. Proto je i zde možné využít výrazu `<propertyExpression>`, ve kterém se definuje, jestli se hodnota úrovně seskupení má vytvořit, nebo nemá (*Ukázka kódu 61 – Dynamická tvorba seskupení*).

```
…
<variable         name="SUM(LB_WE)_1"         class="java.math.BigDecimal"
resetType="Group" resetGroup="dodavatel" calculation="Sum">
  <variableExpression>
    <![CDATA[$F{SUM(LB_WE)}]]>
  </variableExpression>
</variable>
<variable       name="SUM(LB_WE)_1_1"       class="java.math.BigDecimal"
resetType="Group" resetGroup="sklad" calculation="Sum">
  <variableExpression>
    <![CDATA[$F{SUM(LB_WE)}]]>
  </variableExpression>
</variable>
…
<group name="sklad">
…
  <groupFooter>
    <band height="40">
      <textField>
        <reportElement style="ramecek" x="0" y="0" width="540"
        height="20">
          <property name=
          "net.sf.jasperreports.export.xls.row.outline.level.1"
          value="End"/>
          …
  </groupFooter>
</group>
<group name="dodavatel">
  <groupExpression><![CDATA[$F{DKA_KURZ}]]></groupExpression>
  <groupHeader>
    <band height="20">
      <printWhenExpression><![CDATA[$P{15}=="J"]]></printWhenExpression>
        <textField>
          <reportElement style="ramecek" mode="Transparent" x="0" y="0"
          width="270" height="20">
            <property name=
            "net.sf.jasperreports.export.xls.row.outline.level.1"
            value="Body"/>
            …
  </groupHeader>
  <groupFooter>
    <textField>
      <reportElement style="ramecek" x="0" y="0" width="540" height="20">
        <propertyExpression name=
        "net.sf.jasperreports.export.xls.row.outline.level.2">
          <![CDATA[$P{15}=="J"?"End":null]]>
        </propertyExpression>
        …
  </groupFooter>
<group>
…
<detail>
  <band height="25" splitType="Stretch">
    <printWhenExpression><![CDATA[$P{15}=="J"]]></printWhenExpression>
    <textField>
      <reportElement style="ramecek" mode="Transparent" x="0" y="0"
      width="270" height="25">
        <property name=
        "net.sf.jasperreports.export.xls.row.outline.level.2"
```

```
        value="Body"/>
        …
</detail>
```

Ukázka kódu 61 – Dynamická tvorba seskupení

Ukázka výsledné sestavy exportované do Excelu je obsažena v příloze D, *str. 90*.

4. Zhodnocení metodiky tvorby reportů

JasperReports je velmi šikovný a působivý nástroj, navíc spolu s grafickým návrhářem iReport tvoří celek pro pohodlné vytváření grafických sestav. Díky tomuto WYSIWYG editoru se pro spoustu uživatelů stane tvorba reportů zábavou. Nejen že umožňuje vkládání různých grafických prvků do reportu, ale také dovoluje exportovat sestavu do mnoha různých formátů. Možnosti těchto open-sourcových nástrojů neocení pouze ti, co pomocí nich budou sestavy vytvářet, ale také celé podniky, jejichž finanční stránka nebude natolik zatížena.

Dosavadní možnosti tvorby uživatelských výstupů byly tiskové masky, PDF generátor a Excel manager. Tyto nástroje jsou naprogramovány v jádru OR-SYSTEMu pomocí vývojových prostředků MicroFocus Server[3] (Cobol) - serverová část a IDM[4] (ISA Dialog Manager) - klientská část.

4.1. Tiskové masky

Tiskové programy tvoří v OR-SYSTEMu největší skupinu, u které se hlavně předpokládají úpravy masek „na míru“ konkrétního uživatele. V zásadě je můžeme rozdělit na tři hlavní skupiny:

1. Programy, které provádějí výpočty a výpisy z již existujících dat - např. výpis z katalogu zásob, ale také skladová regleta nebo tisk kupní smlouvy.
2. Programy, jejichž spuštěním dochází kromě sestavení obvykle složitějšího tiskového výstupu také k zásahu do datových souborů (vytváření nových položek a aktualizace existujících) - např. tvorba dodacího listu vč. faktury. V rámci takových programů dochází i k několikanásobnému volání jiných programů nebo podprogramů.

[3] více na www.microfocus.com
[4] více na www.isa.de

3. Programy, kterými zadáváme provedení nějaké operace s daty; tiskový výstup (obvykle protokol) má jen kontrolní funkci - např. převod fakturace do statistik, přecenění, sestavení kalkulací atd.

Pro každý blok masky, který obsahuje nastavení pevných textů, může být vytvořen „párový“ blok, ve kterém jsou umístěny atributy pro zvýrazňování textu, resp. rámečky. Vztah mezi oběma „párovými“ bloky je pevně dán jejich jménem, např.: textovému bloku U2 vždy odpovídá grafický blok 2U (viz příloha E, *str. 91*).

Formáty tisku mohou být: DOC, RTF, XLS.

4.2. PDF generátor

V dokumentu PDF jsou v OR-SYSTEMu podporovány grafické prvky jako textové pole, obrázek JPG, zvýrazňovací obdélník, čárový kód a čára. Toto už samo o sobě vypovídá, že by JasperReports byl vhodnějším nástrojem pro tvorbu grafických reportů z důvodu většího množství prvků, které lze do sestavy vkládat. Zde se zadává několik hodnot pro definici výšky, šířky, barvy atd., kdežto grafický návrhář iReport umožňuje umisťovat pole metodou „Drag and drop“ tam, kam je potřeba a také nastavovat u těchto polí styl písma, barvu, velikost atd. prostřednictvím vlastností. Oproti tomu v PDF generátoru se například barvy určují na základě dokumentu barvy.pdf, kde jsou jednotlivé barvy definovány (např. 0 - černá, 86 – žlutá…). Příklad obdélníku, který je uvedený řetězcem $RECT a následuje zápis pevného formátu, je definován v ukázce kódu – *Obdélník*.

```
* $RECT  obdelniky      slou rade sirk vysk se ba
* $R1M   obdelnik1      01.0 02.0 40.0 01.0 80 00
* $R1M   obdelnik2      01.0 12.0 40.0 10.0 90 00
```

Ukázka kódu 62 – Obdélník

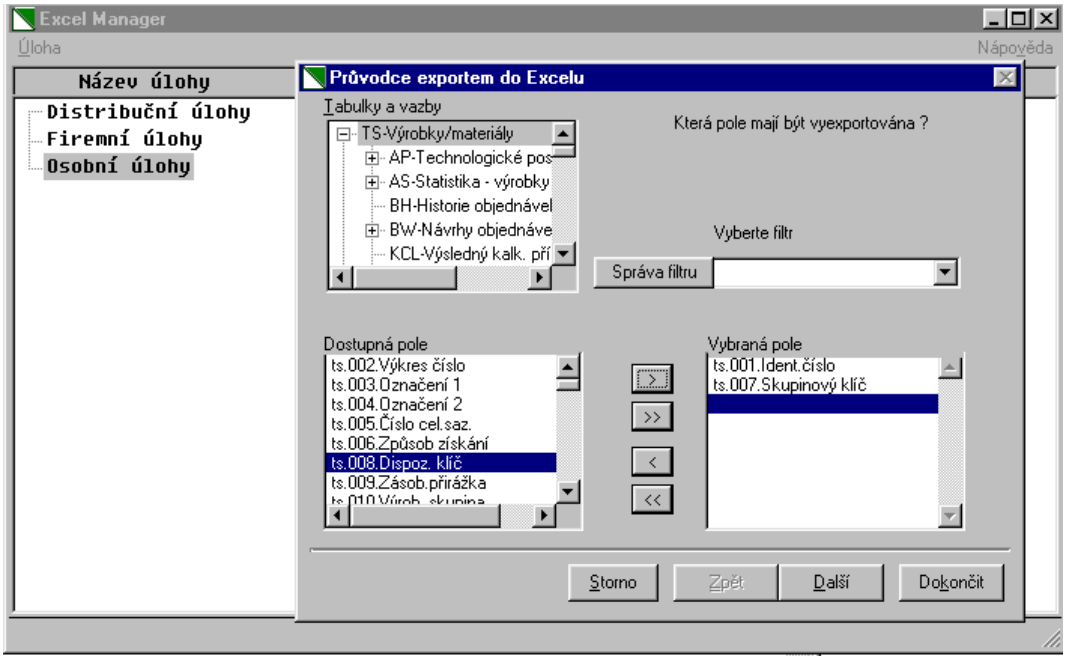
4.3. Excel Manager

Excel Manager je nástroj pro klíčové uživatele OR-SYSTEMu, kteří chtějí vytvářet uživatelsky definované výstupy z OR-SYSTEMu do tabulkových procesorů jako je

MS Excel, OpenOffice, 602PC Suite, nebo do formátu XML. Do sestav lze vkládat grafy a kontingenční tabulky.

Program umožňuje:

1. Tvorbu firemních úloh i úloh pro jednotlivé uživatele, jejich trvalé uložení
2. Možnost aktualizace a rušení existujících úloh
3. Spojování více datových tabulek prostřednictvím klíčových slov v definovaném rozsahu
4. Výběr libovolných polí z každé definované datové tabulky
5. Převod dat dle vytvořené struktury do Excelu



Obrázek 16 – Excel Manager
Zdroj: Printscreen - OR-SYSTEM

4.4. Shrnutí

Tyto nástroje mají jistě své výhody i nevýhody, stejně jako JasperReports. Nýbrž již významnou výhodou JasperReports oproti těmto nástrojům je, že je to jeden jediný nástroj a umožňuje výstupy do nejrůznějších formátů. Kromě vkládání grafických prvků do sestavy, kterých JasperReports nabízí veliké množství, dodává sestavám lepší přehlednost a vizualizaci, a dalo by se říci, že díky grafickému návrháři se sestavy tvoří

velmi snadno a rychle. Podle mého názoru je pro uživatele také velmi přínosné, že iReport umožňuje vytvářet vzhled výstupu tak, jak bude vypadat při tisku, což s předchozími nástroji nebylo při tvorbě grafických výstupů z dat OR-SYSTEMu možné.

Jako přednost tohoto nástroje bych dále vyzdvihla možnost výstupu do formátu OpenOffice, OpenDocumentText (ODT). Neboť, jak se praví v časopise Open Source o zavedení OpenOffice ve firmě Sapeli, a.s.: *„Na rozdíl od Microsoft Office k němu není nutné kupovat licence, jeho použití ve firmě tak může být bezplatné, i když bude nasazen na stovkách či tisících počítačů.“* [13, s. 2] Dalším dobrým důvodem je multiplatformnost, jelikož třeba například ve firmě Sapeli, a.s. testují Linux i na klientských stanicích, přičemž Microsoft Office linuxovou verzi vůbec nenabízí.

Pravdou je, že spousta uživatelů, kteří už mají předchozí způsoby tvorby reportů zažité, nové možnosti mohou odmítat z důvodu nezájmu o nové věci. Ale o tom také rozhoduje kvalita příručky k tomuto nástroji a samotná školení věnovaná uživatelům. Na druhou stranu JasperReports nahradí svou funkcionalitou značnou část uživatelské práce, na kterou je potřeba kancelářský balík. [13]

Závěr

Tato práce umožňuje získat informace o tvorbě tiskových sestav pomocí využití reportovacího nástroje JasperReports. První část práce se zabývá objektově relačním mapování, jelikož metodika tvorby reportů byla navržena s využitím nástroje Hibernate, který objektově relační mapování umožňuje. Samotná metodika je obsažena v části druhé. Vlastní návrh metodiky byl aplikován na datech uživatelů informačního systému OR-SYSTEM, což je součástí třetího bodu této práce. Metodiku, která byla navržena tak, aby byla pro uživatele co nejsrozumitelnější a zajišťovala jim co největší komfort, jsem měla možnost také ověřit při školení některých zákazníků, ale i zaměstnanců firmy OR-CZ, spol. s r.o., pro které jsem vytvářela příručku k této metodice. Poslední část práce zahrnuje možnosti nástroje JasperReports v porovnání s dosavadními řešeními tvorby reportů.

Jelikož jsou JasperReports i grafický návrhář iReport open-sourcové nástroje, které poskytují spoustu možností při návrhu sestavy, jež jsou v této práci popsány, jsou přínosem nejen pro uživatele samotných nástrojů, kterým ušetří čas, ale také pro firmy, kterým se uleví z hlediska finančního. Aby byly výsledky práce uživatelů opravdu co nejefektivnější, musí se s těmito nástroji nejdříve naučit pracovat, což lze zařídit kvalitními školeními a příručkami.

Poněvadž je JasperReports velice mocný nástroj, skýtá spousty dalších možností, kterých lze v budoucnu využít. Příkladem může být webová aplikace JasperReports Server, která kupříkladu umožňuje přistupovat k sestavám přes webové rozhraní, zasílat je e-mailem nebo exportovat do formátů, které JasperReports nabízí. Je tedy v moci vývojářů nacházet lepší a efektivnější řešení.

Seznam použité literatury

Citace

[1] HOEBER, M., S. HOMMEL, I. RABINOVITCH, T. RISSER, J. ROYAL a S. ZAKHOUR. *Java 6: výukový kurz*. 1. vyd. Brno: Computer Press, a.s., 2007. ISBN 978-80-251-1575-6.

[2] HERNANDEZ, Michael J. a John. L. Viescas. *Myslíme v jazyku SQL: knihovna programátora.* Praha: Grada Publishing, a.s., 2004. ISBN 80-247-0899-X.

[3] LONEY, K. a M. THERIAULT. *Mistrovství v Oracle: kompletní průvodce tvorbou, správou a údržbou databáze.* 1. vyd. Praha: Computer Press, 2002. ISBN 80-7226-635-7.

[4] SOLAŘ, T. *Oracle Database 11g: hotová řešení.* 1. vyd. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2886-2.

[5] Hibernate (Java). In: *Wikipedia: otevřená encyklopedie* [online]. Los Angeles (California): Wikimedia Foundation, 2001-, strana naposledy edit. 2012-04-13, 14:45 [vid. 2012-04-27, 19:00]. Anglická verze. Dostupné z: http://en.wikipedia.org/wiki/Hibernate_%28Java%29

[6] BÖCK, H. *Platforma NetBeans: Podrobný průvodce programátora.* 1. vyd. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-3116-9.

[7] GNU Lesser General Public License. In: *Wikipedia: otevřená encyklopedie* [online]. Los Angeles (California): Wikimedia Foundation, 2001-, strana naposledy edit. 2011-12-12, 16:38 [vid. 2012-04-27, 19:05]. Česká verze. Dostupné z: http://cs.wikipedia.org/wiki/LGPL

[8] HEFFELFINGER, David R. *JasperReports 3.5 for Java developers.* 1st ed., Birmingham: Packt Publishing Ltd., 2009. ISBN 978-1-847198-08-2.

[9] PROCHÁZKA, T. *Výukový kurz použití JasperReports frameworku* [online]. Praha: České vysoké učení technické v Praze, 2006 [vid. 2012-04-27, 20:08]. Dostupné z: http://atom.mamto.cz/projekty/skola/SWT/JasperReports_tutorial.pdf

[10] JEŽEK, K. *JasperReports - tisk v Javě* [online]. Plzeň: Západočeská univerzita v Plzni, 2007, strana naposledy edit. 2007-06-15 [vid 2012-04-27, 20:54]. Dostupné z: http://www.java.cz/article/6795

[11] TOFFOLI, G. *Getting Started* [online]. JasperSoft Corporation, 2001-, [vid. 2012-04-27, 21:43]. Dostupné z: http://jasperforge.org/website/ireportwebsite/IR%20Website/ir_getting_started.html?header=project&target=ireport

[12] *OR-SYSTEM* [online]. Moravská Třebová: OR-CZ, spol. s r.o. [vid. 2012-04-29, 22:06]. Dostupné z: http://www.orcz.cz/www/www-new.nsf/0/06F9C1E394F278B2C12577A00029502C?OpenDocument

[13] BÍBR, I. Open source & praxe. *SAPELI – dveře a zárubně s Linuxem.* Olomouc: Liberix, o.p.s., 2012, roč. 2, č. 1. ISSN 1804-8560.

Bibliografie

HEUDECKER, N. and P. PEAK. *Hibernate Quickly.* Greenwich: Manning Publications Co., 2006. ISBN 1-932394-41-9.

KISZKA, B. *1001 tipů a triků pro jazyk Java*. 1. vyd. Brno: Computer Press, a.s., 2009. ISBN 978-80-251-2467-3.

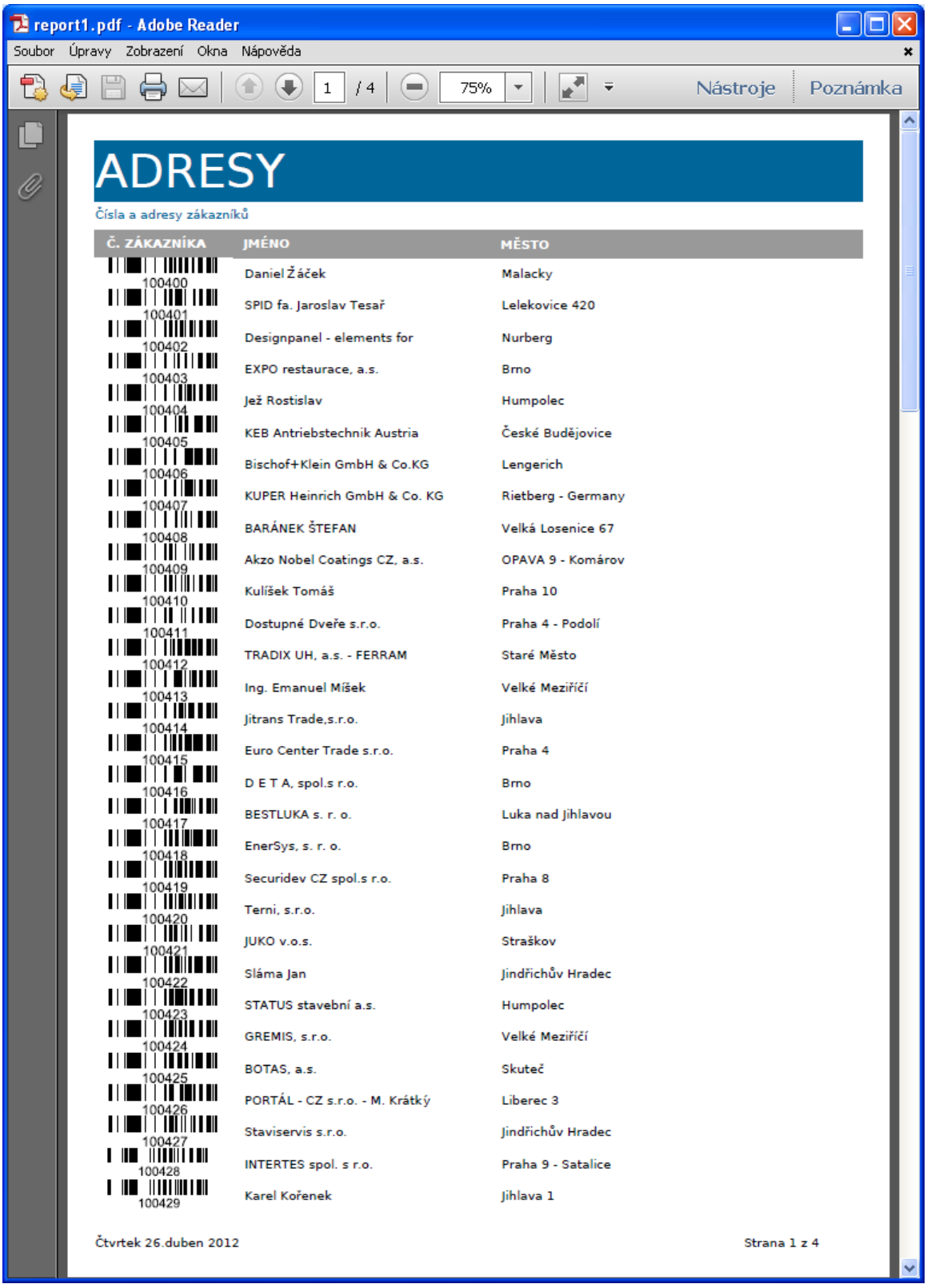
VALENTIN R., J. STEINER. *Access 97: Kompletní kapesní průvodce*. Praha: Grada Publishing, spol. s r.o., 1998. ISBN 80-7169-618-8.

VIRIUS M. *Java pro zelenáče*. Praha: Neocortex spol. s r.o., 2001. ISBN 80-902230-9-5.

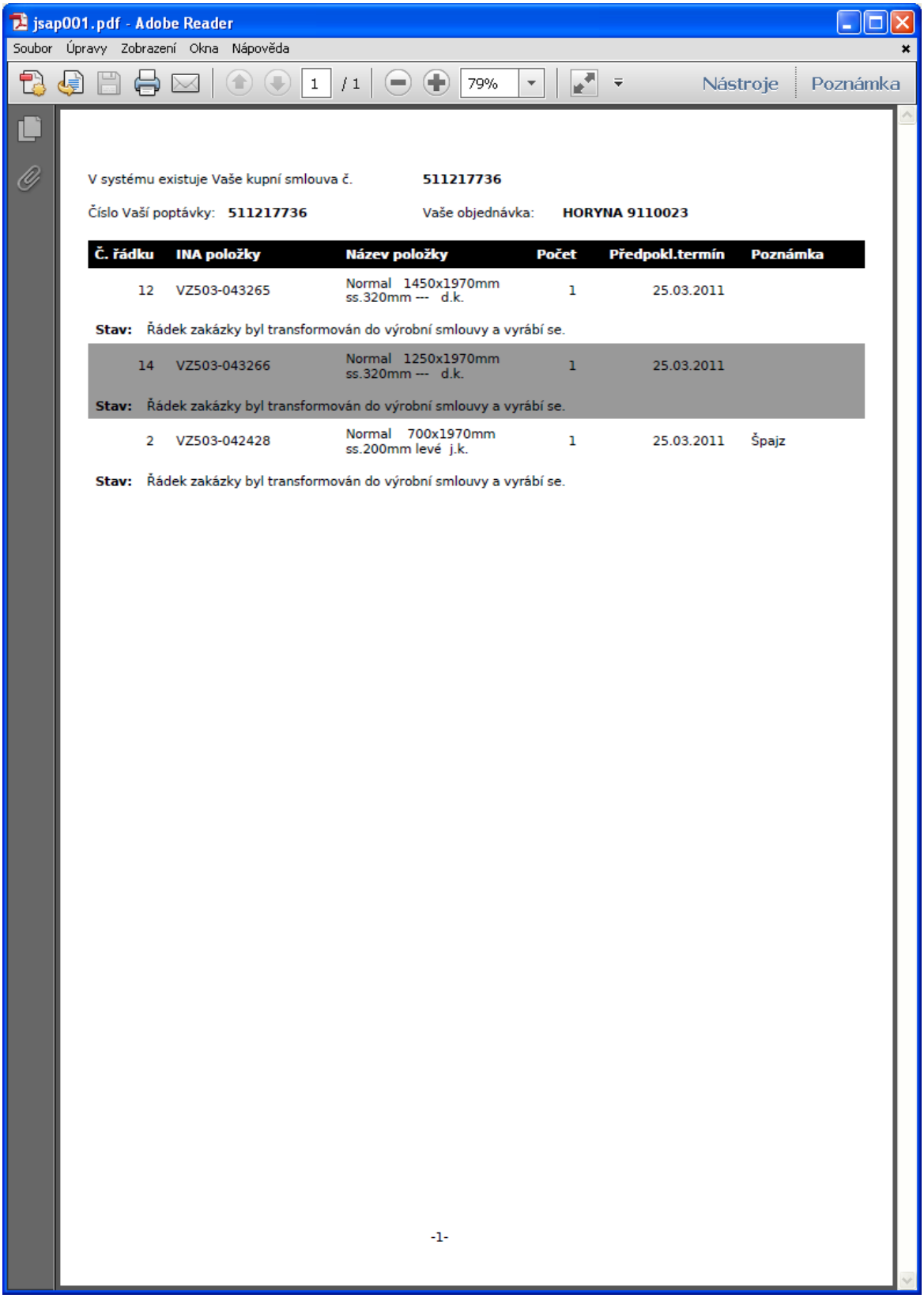
Seznam příloh

Přílohy

Příloha A Ukázková sestava ve formátu PDF



Příloha B Stav zakázky – výsledná sestava exportovaná do formátu PDF



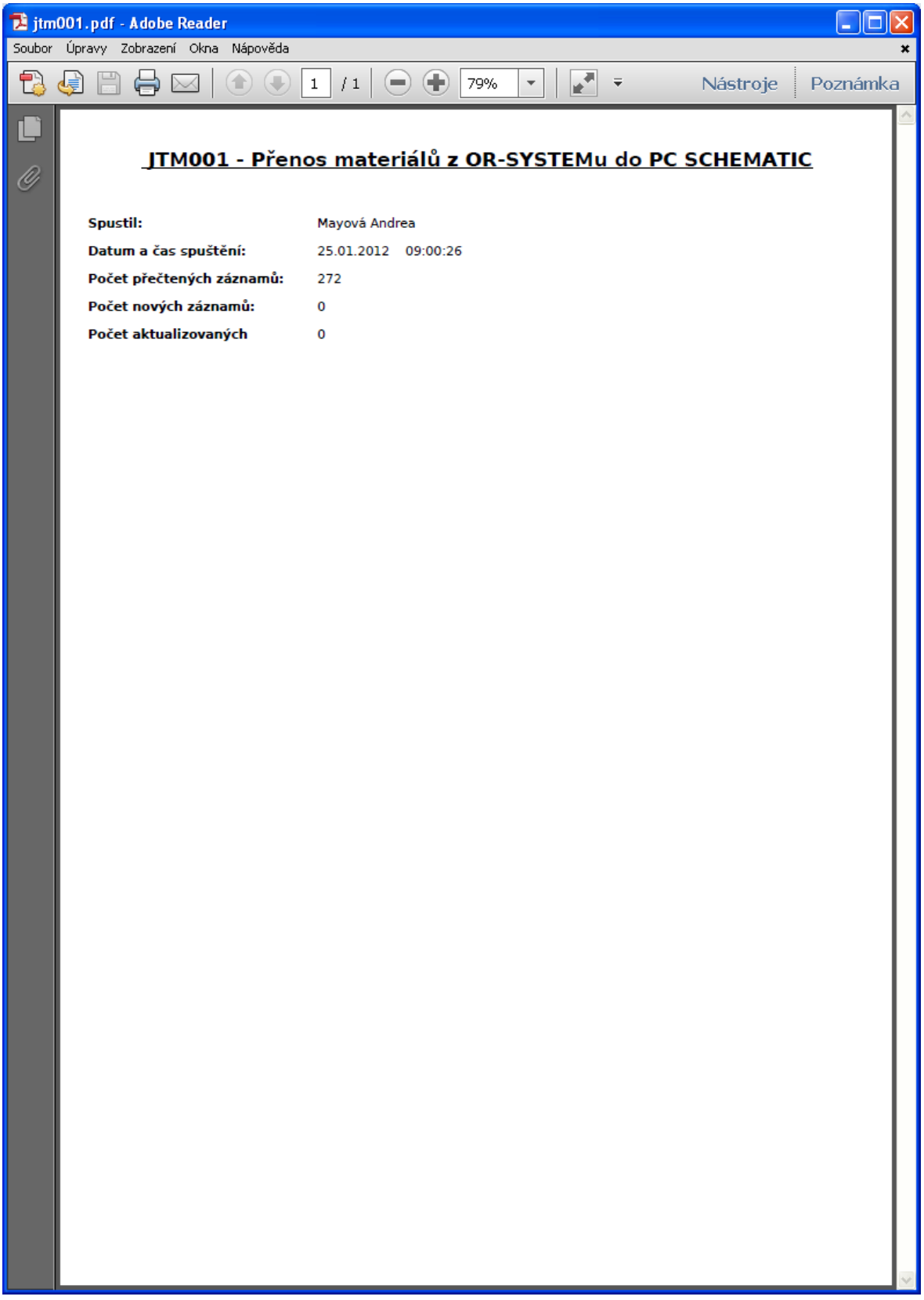
V systému existuje Vaše kupní smlouva č. **511217736**

Číslo Vaší poptávky: **511217736** Vaše objednávka: **HORYNA 9110023**

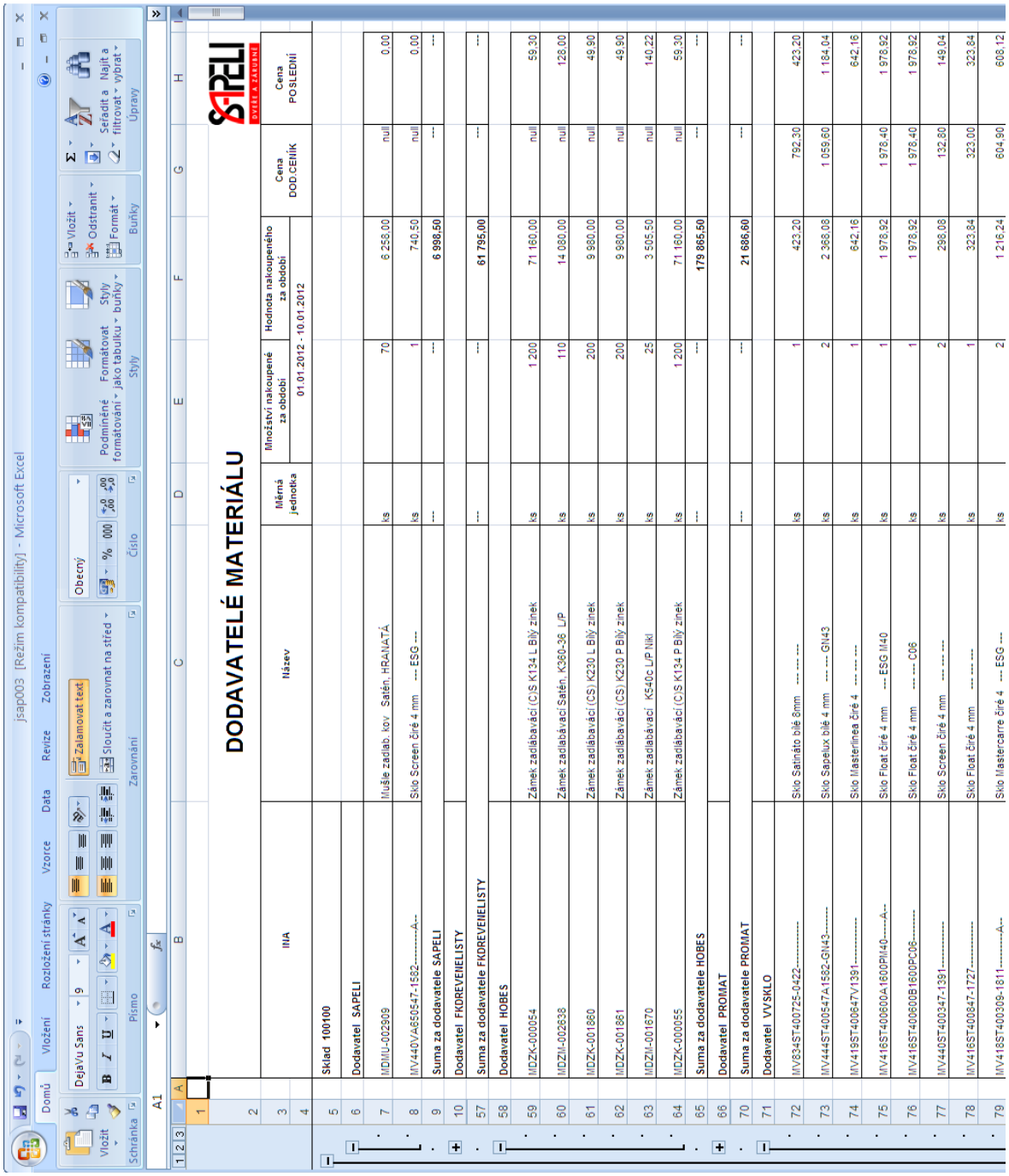
Č. řádku	INA položky	Název položky	Počet	Předpokl.termín	Poznámka
12	VZ503-043265	Normal 1450x1970mm ss.320mm --- d.k.	1	25.03.2011	
Stav:	Řádek zakázky byl transformován do výrobní smlouvy a vyrábí se.				
14	VZ503-043266	Normal 1250x1970mm ss.320mm --- d.k.	1	25.03.2011	
Stav:	Řádek zakázky byl transformován do výrobní smlouvy a vyrábí se.				
2	VZ503-042428	Normal 700x1970mm ss.200mm levé j.k.	1	25.03.2011	Špajz
Stav:	Řádek zakázky byl transformován do výrobní smlouvy a vyrábí se.				

-1-

Příloha C Přenos materiálů - výsledná sestava exportovaná do formátu PDF



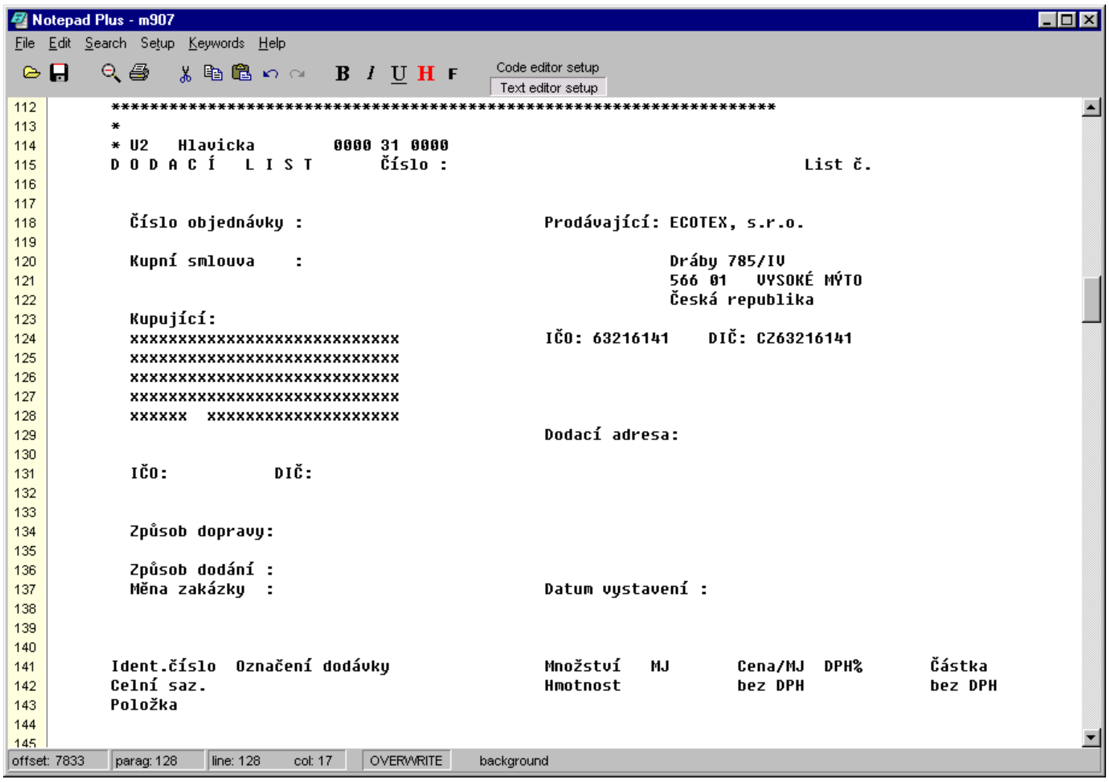
Příloha D Sklady, dodavatelé – výsledná sestava exportovaná do Excelu



DODAVATELÉ MATERIÁLU

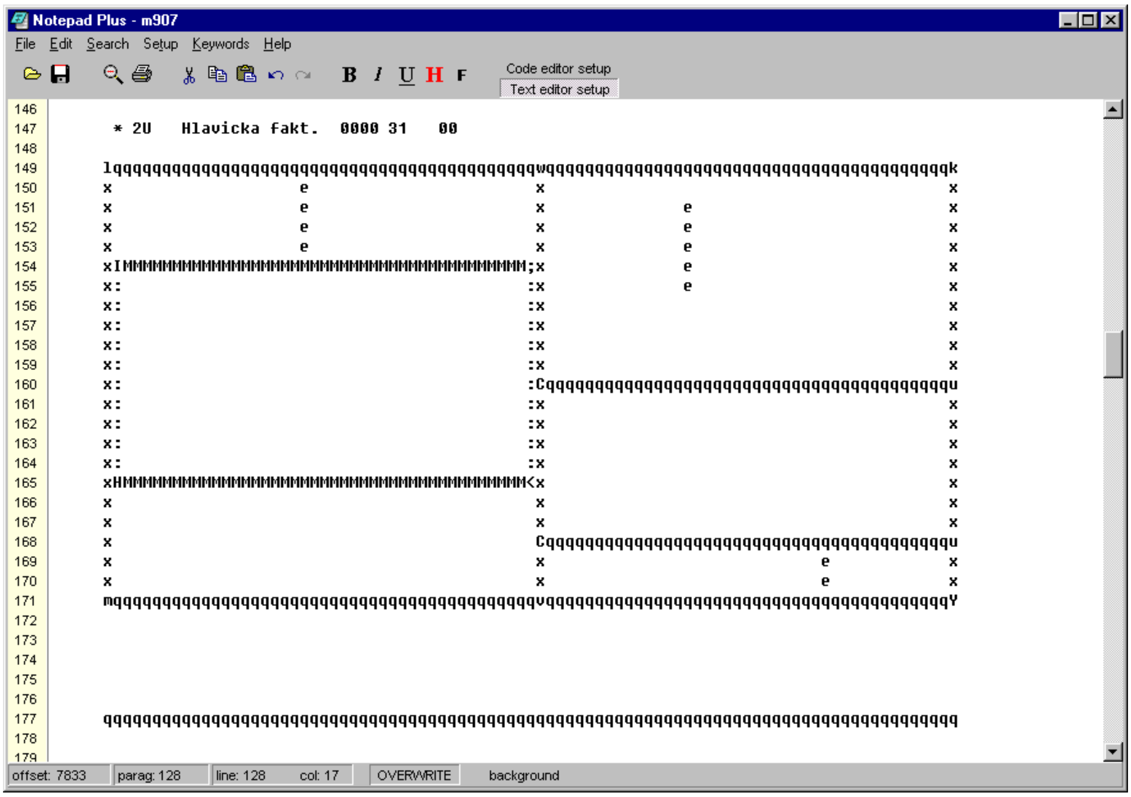
IIIA	Název	Měrná jednotka	Množství nakoupené za období 01.01.2012 - 10.01.2012	Hodnota nakoupeného za období	Cena DOD.CENÍK	Cena POSLEDNÍ
Sklad 100100						
Dodavatel SAPELI						
MDMU-002909	Mušle zadlab. kov Satén, HRANATÁ	ks	70	6 258,00	null	0,00
MV440VA650547-1582----------A--	Sklo Screen Čiré 4 mm --- ESG ---	ks	1	740,50	null	0,00
Suma za dodavatele SAPELI		---	---	6 998,50	---	---
Dodavatel FKDREVENELISTY						
Suma za dodavatele FKDREVENELISTY		---	---	61 795,00	---	---
Dodavatel HOBES						
MDZK-000054	Zámek zadlabávací (C)S K134 L Bílý zinek	ks	1 200	71 160,00	null	59,30
MDZM-002638	Zámek zadlabávací Satén, K360-38 L/P	ks	110	14 080,00	null	128,00
MDZK-001860	Zámek zadlabávací (CS) K230 L Bílý zinek	ks	200	9 980,00	null	49,90
MDZK-001861	Zámek zadlabávací (CS) K230 P Bílý zinek	ks	200	9 980,00	null	49,90
MDZM-001670	Zámek zadlabávací K540c L/P Niki	ks	25	3 505,50	null	140,22
MDZK-000055	Zámek zadlabávací (C)S K134 P Bílý zinek	ks	1 200	71 160,00	null	59,30
Suma za dodavatele HOBES		---	---	179 865,50	---	---
Dodavatel PROMAT						
Suma za dodavatele PROMAT		---	---	21 686,60	---	---
Dodavatel VVSKLO						
MV834ST400725-0422----------	Sklo Satinato bílé 8mm --- --- ---	ks	1	423,20	792,30	423,20
MV444ST400547A1582-GN43-------	Sklo Sapelux bílé 4 mm --- --- GN43	ks	2	2 368,08	1 059,60	1 184,04
MV419ST400647V1391----------	Sklo Masterlinea Čiré 4 --- --- ---	ks	1	642,16		642,16
MV416ST4006004A1600PM40------A--	Sklo Float Čiré 4 mm --- ESG M40	ks	1	1 978,92	1 978,40	1 978,92
MV416ST400600B1600PC06--------	Sklo Float Čiré 4 mm --- --- C06	ks	1	1 978,92	1 978,40	1 978,92
MV440ST400347-1391----------	Sklo Screen Čiré 4 mm --- --- ---	ks	2	298,08	132,80	149,04
MV416ST400847-1727----------	Sklo Float Čiré 4 mm --- --- ---	ks	1	323,84	323,00	323,84
MV418ST400309-1811---------A--	Sklo Mastercarre Čiré 4 --- ESG ---	ks	2	1 216,24	604,90	608,12

Příloha E Tiskové masky – záhlaví dokladu



```
*********************************************************************
*
* U2   Hlavicka        0000 31 0000
D O D A C Í   L I S T       Číslo :                                     List č.


  Číslo objednávky :                         Prodávající: ECOTEX, s.r.o.

  Kupní smlouva    :                                      Dráby 785/IV
                                                          566 01   VYSOKÉ MÝTO
                                                          Česká republika
  Kupující:
  xxxxxxxxxxxxxxxxxxxxxxxxxxxx               IČO: 63216141     DIČ: CZ63216141
  xxxxxxxxxxxxxxxxxxxxxxxxxxxx
  xxxxxxxxxxxxxxxxxxxxxxxxxxxx
  xxxxxxxxxxxxxxxxxxxxxxxxxxxx
  xxxxxx  xxxxxxxxxxxxxxxxxxxx
                                             Dodací adresa:

  IČO:           DIČ:


  Způsob dopravy:

  Způsob dodání :
  Měna zakázky  :                            Datum vystavení :



Ident.číslo  Označení dodávky                Množství   MJ       Cena/MJ  DPH%       Částka
Celní saz.                                   Hmotnost            bez DPH             bez DPH
Položka
```

Blok U2 (nastavení počtu řádků hlavičky; pevné texty, které se budou do záhlaví vždy tisknout; xxxxx - pomocná matice, která bude na tiskovém výstupu přepsána skutečnými údaji)



```

 * 2U    Hlavicka fakt.  0000 31   00

lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqwqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x                    e                       x                                          x
x                    e                       x               e                          x
x                    e                       x               e                          x
x                    e                       x               e                          x
xIMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM;x               e                          x
x:                                          :x               e                          x
x:                                          :x                                          x
x:                                          :x                                          x
x:                                          :x                                          x
x:                                          :x                                          x
x:                                          :Cqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqu
x:                                          :x                                          x
x:                                          :x                                          x
x:                                          :x                                          x
x:                                          :x                                          x
xHMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM<x                                          x
x                                            x                                          x
x                                            x                                          x
x                                            Cqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqu
x                                            x                             e            x
x                                            x                             e            x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqvqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqY





qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
```

Blok 2U (nastavení atributů pro tisk záhlaví dodacího listu)